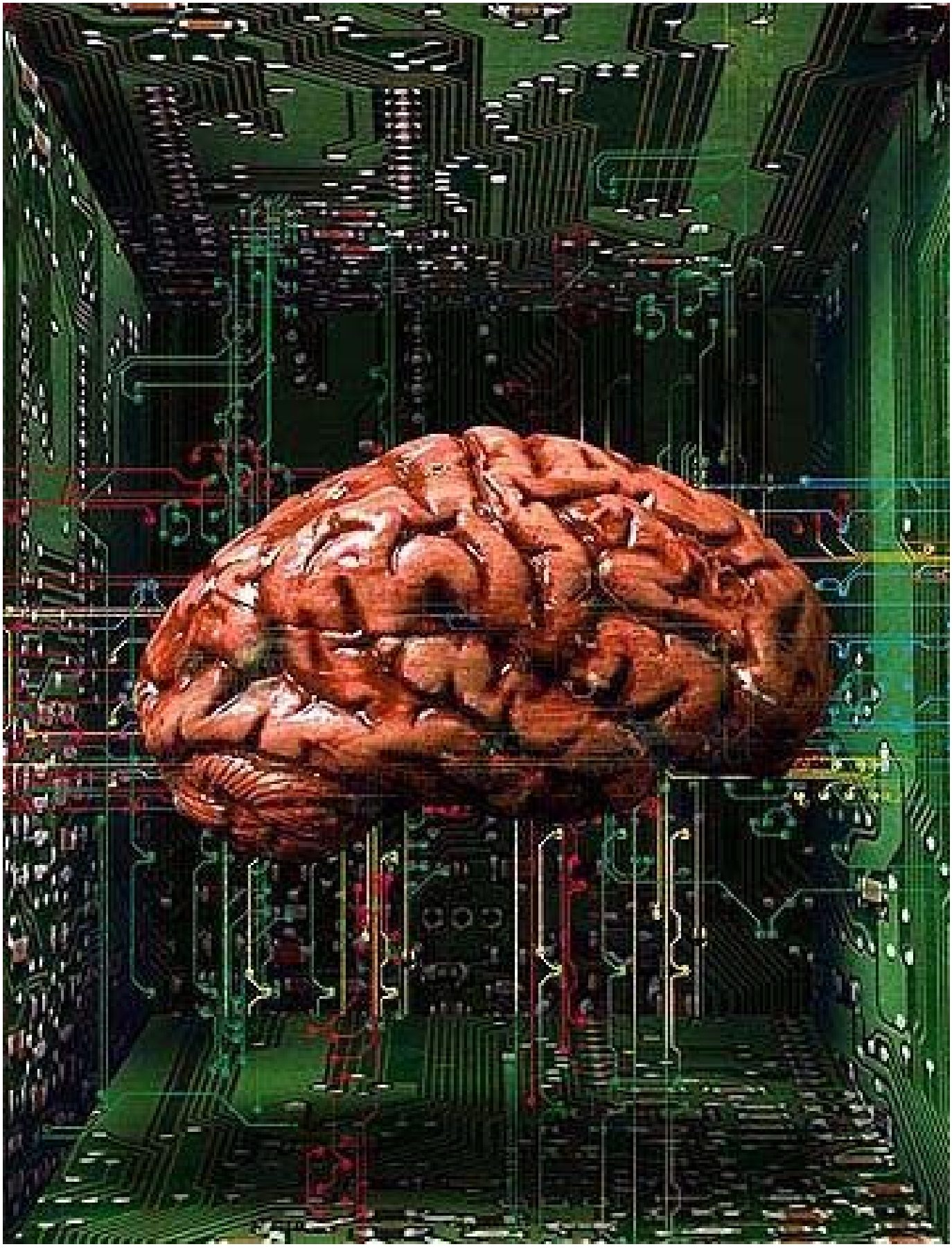
Brain-Computer Interfaces



BRAIN-COMPUTER INTERFACES

DESIGN AND IMPLEMENTATION OF AN ONLINE BCI SYSTEM FOR THE CONTROL IN GAMING APPLICATIONS AND VIRTUAL LIMBS

vorgelegt von

Diplom - Informatiker Roman Krepki

von der Fakultät IV — Elektrotechnik und Informatik der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften — Dr. rer. nat. —

genehmigte Dissertation

Promotionsausschuss:

- Vorsitzender: **Prof. Dr. rer. nat. Klaus Obermayer**
- Gutachter: **Prof. Dr. Ing. Stefan Jähnichen**
  Prof. Dr. rer. nat. Klaus-Robert Müller

Tag der wissenschaftlichen Aussprache: 26. November 2004.

Berlin 2004
D 83

TO MY GRANDFATHER

GRIGORIJ K. GORLOV

✵ **December, 25th, 1919.**
✞ **August, 14th, 2003.**

ABSTRACT

This dissertation aims to describe work carried out at the Fraunhofer Institute for Computer Architecture and Software Technology (FhG-FIRST), in particular at the research group for Intelligent Data Analysis (IDA), within the project "Brain-Computer Interface" (BCI). The goal of that project is to design and develop a hardware and software system that is capable of transforming, in real-time, electroencephalographic (EEG) signals (signals retrieved in a non-invasive way from surface electrodes placed over the user's head) into specific commands such that the user gains reliable control over a computer application or a device.

In this dissertation, control over a computer application will be represented with "Brain-Gaming", i.e. simple computer games like Ping-Pong, Pacman or Tetris. To this end, substantial considerations were made on the design and realization of a communication interface and its corresponding protocol. A further important component of every control operation is its strategy and the control alphabet, i.e. the command set. For this purpose, several control strategies were developed, implemented and proved experimentally in different scenarios. The most important aspect of the design and development of these interfaces turned out to be their flexibility. Control over a device could include the steering and moving of a wheel-chair for paralyzed patients or the gaining of control over an arm or foot prosthesis for patients with amputated limbs. The latter was realized here as a computer-based simulation of a virtual limb (e.g. arm), such that it can be tested in future experiments on patients with amputated limbs.

In contrast to the "Brain-Gaming" experiments, where the player was equipped with an additional communication channel (one that exists independently of normal communication channels of the human neuromuscular system), the experiments with patients did not require an ultra-fast recognition of the intended movement; i.e. the command signal for a simulated movement can be recognized after the phantom movement is initialized and performed. In experiments with feedback scenarios, which can be resembled as competitive games, several aspects of ultra-fast movement detection could be investigated with reaction tests. This opens new perspectives for the execution of preventive actions in time-critical applications.

This dissertation, and the development and implementation of the prototype, is based on well-founded insights into human neurophysiology; one chapter will deal exclusively with these insights.

Moreover, a special chapter of this dissertation will also describe the development and implementation of an online prototype of the BBCI system (Berlin Brain-Computer Interface) from the software engineering viewpoint. A number of bio-feedback modules, for gaming and

for rehabilitation purposes, were developed within this work and will be presented in detail. Special attention was paid to the influence of the online bio-feedback on the user's behavior.

ZUSAMMENFASSUNG

Die vorliegende Dissertation beschreibt Ergebnisse der Arbeit, durchgeführt am Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik (FIRST), insbesondere bei der Forschungsgruppe für Intelligente Datenanalyse (IDA), im Rahmen des Projektes „Brain-Computer Interface“ (BCI). Das angestrebte Ziel des aktuell laufenden Projektes ist es, ein Hardware- und Software-System zu entwerfen und zu entwickeln, das in der Lage ist elektroencephalographische (EEG) Signale (gewonnen auf eine nicht-invasive Art, mit Hilfe der Oberflächenelektroden, die über dem Kopf des Benutzers angebracht sind), in Echtzeit in spezielle Kommandos umzuwandeln, so dass für den Probanden eine verlässliche Steuerung einer Computeranwendung, bzw. eines Gerätes ermöglicht wird.

Die Steuerung einer Computeranwendung soll im Rahmen dieser Arbeit in Form von einfachen Computerspielen (Ping-Pong, Pacman, Tetris) repräsentiert werden – im Weiteren bezeichnet als „Brain-Gaming“. Hierzu sind fundierte Überlegungen zum Entwurf und Realisierung einer Kommunikationsschnittstelle und des zugehörigen Protokolls angestellt worden. Ein weiterer wichtiger Bestandteil jeder Steuerung ist deren Strategie und der Befehlssatz der Anwendung. So wurden mehrere Strategien entwickelt, implementiert und in verschiedenen Szenarien experimentell erprobt. Die Flexibilität der Steuerungsschnittstelle stellte sich als einer der wichtigsten Aspekte beim Entwurf und der Entwicklung von Rückkopplungsanwendungen. Bei der Steuerung eines Gerätes kann es sich um das Lenken und Bewegen eines Rollstuhls, z.B. für querschnittsgelehmte Patienten, oder um das Bewegen einer Arm-, bzw. Beinprothese für Patienten mit amputierten Extremitäten handeln. Dies wurde vorerst als Simulation einer Extremität (Arm) auf dem Computer-Bildschirm realisiert, so dass es in zukünftigen Experimenten an bedürftigen Patienten getestet werden kann.

Im Gegensatz zu „Brain-Gaming“ Experimenten, bei denen der Spieler mit einem zusätzlichen Kommunikationskanal (der unabhängig von anderen normalen Kanälen des menschlichen neuromuskulären Systems ist) ausgestattet wird, stellen die Experimente an Patienten keine Anforderung an die ultra-schnelle Erkennung der Bewegungsabsicht; So kann das Steuersignal zur Ausführung einer simulierten Bewegung auch nach dem Auslösen der eigentlichen Phantombewegung, sogar nach deren Ausführung, erkannt werden. In Experimenten mit Feedback-Szenarien, die kompetitiven Spielen ähneln, können verschiedene Aspekte der ultraschnellen Erkennung einer Bewegungsintension mit Hilfe von Reaktionstests untersucht werden. Dieses eröffnet neue Perspektiven bei der Ausführung von Präventivmaßnahmen in zeitkritischen Anwendungen.

Diese Dissertation, sowie die Entwicklung und Implementierung des Prototyps basiert auf fundierten Erkenntnissen der Neurophysiologie; Ein ausgewähltes Kapitel dieser Dissertation verschafft deshalb tieferen Einblick in die menschliche Neurophysiologie.

Ferner, beschreibt ein gesondertes Kapitel dieser Dissertation die Entwicklung und Implementierung eines online Prototyps – des BBCI-Systems (Berlin Brain-Computer Interface) – in dessen Einzelkomponenten und der Gesamtheit aus dem Sichtpunkt der Softwaretechnik. Im Rahmen der Arbeit wurden mehrere Rückkopplungsmodule (Bio-Feedback), sowohl spielerischen Charakters, als auch zu Rehabilitationszwecken entwickelt, die hier im Detail vorgestellt werden. Mit besonderer Aufmerksamkeit wurde der Einfluss des online Bio-Feedbacks auf den Probanden untersucht.

ACKNOWLEDGMENTS

Ganz elend ist keiner trotz üblen Siechtums:
Den Einen beseligt ein Sohn;
Den Zweiten – Verwandtschaft;
Sein Wohlstand – den Dritten,
Den Vierten – ein würdiges Werk
Edda (Liedersammlung 9th – 12th century A.D.)

My first acknowledgment is addressed to *Dr. habil. Martin Stetter* for his ability to inspire me for the topic of Brain-Computer Interfacing. He, for one, presented the BCI research field when I was looking for an interesting thesis. From this moment on my heart was set on this promising project. I imparted my inspiration with my former colleague, *Dipl.-Math. Hendrik Purwins*, who referred me to some of his friends who were to start working on a BCI project. He brought me to the Fraunhofer-FIRST, where, lo and behold, I re-encountered my desired BCI project. Thus, good advice is not always hard to come by! I am very grateful to Hendrik for that lead.

The head of the laboratory for Intelligent Data Analysis (IDA) at the Fraunhofer Institute for Computer Architecture and Software Technology (FhG-FIRST), *Prof. Dr. Klaus-Robert Müller* happened to be looking for a computer scientist and software engineer to design and realize a prototype of a BCI system and hired me for this position. I am very grateful to him for his faith in me and I hope to have fulfilled his expectations. Klaus almost always found the time to speak with his colleagues despite having a busy agenda, being simultaneously involved in many other projects, as well as being responsible for the overall team and project management. I appreciated in particular his many visionary comments, if not to say too optimistic fervor. My most special thanks are dedicated to another leading figure of the BBCI project, *Dr. Benjamin Blankertz*, who often met emerging problems with a sense of healthy realism and who had an open ear to most suggestions and proposed solutions.

I also dedicate special acknowledgment to the co-director of the BBCI project, *Prof. Dr. Gabriel Curio*, who is responsible for the neurological part of the project. He is an inquisitive researcher and also a very nice person. After every discussion with him, I was invariably enriched with new ideas and a bulk of important neurophysiological knowledge that he was able to summarize into a few sentences and transmit in an understandable language. Moreover,

he was extremely reliable and I appreciated that a deadline announced by Gabriel was one that could actually be counted on!

I had very fruitful discussions with *Dr. Ricardo Vigario*, with whom I shared an office when I first came to FIRST. He helped me to delve into the hard scientific work and the more complicated aspects of the BCI project. I would also like to thank *Dipl.-Math Guido Dornhege* for the good collaboration and the many constructive discussions throughout the project. It is well-known that mathematicians and computer scientists often disagree on certain issues of software engineering. Yet Guido was sometimes able to be "won over" for the computer scientist approach that includes software design activities such as working out proper specifications of tasks and solutions, writing an understandable code and instant commenting. These activities do not bring advantages at the first sight, but prove convenient at a later point in time.

Many other people were involved in the BCI project who worked very hard to achieve our common purpose: a successful realization of innovative ideas. I would like to thank *Dr. Jens Kohlmorgen*, *Matthias Krauledat*, *Christin Schäfer*, *Nicolas Heess*, *Thorsten Zander*, *Philipp Seuring* and *Amin Halihel* for doing a great job within the scope of the project. Often people from the UKBF helped us to perform experiments, explained difficult medical or neurological processes, or were simply at the right place when I needed help with problems. Many thanks also to *Dr. Florian Losch* and *Dr. Volker Kunzmann*. The time I spent at FIRST was very enriching and inspiring for me in all respects. This was not only due to its excellent organization and flat hierarchy, but also to the friendly, enjoyable and ingenious members of the team. Thank You IDA for the good time!

On a private note I would like to thank foremost my mother, *Ljudmila Neumann*, and my stepfather, *Heinz-Peter Schneider*, who stood by me and supported me through the good and bad times with insightful advice or an interesting book, yet most often with a delicious meal. I thank very much my girlfriend *Natalja* for bearing with me during the difficult writing-up period and for providing me with all the things that make life beautiful. I would also like to thank all my relatives for their faith in my success; I hope to have fulfilled their expectations.

To all of you, thank you very much for your unlimited help and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I —

INTRODUCTION

Deine Absicht erst gibt deinem Werke seinen Namen.
Ambrosius Aurelius de Milano (340-397 A.D.)

In this introductory Chapter I will describe the topics addressed in the thesis and provide readers not familiar with Brain-Computer Interfacing an insight into this relatively new research field. A brief overview of how this thesis is organized will then follow with suggestions for different reading strategies. Section 2 is addressed to "novices" who have never been exposed to the research and development in the field of Brain-Computer Interfacing. With short descriptions of BCIs proposed by other groups, I will then give a summary of the state-of-the-art in this field; however, I do not claim or guarantee completeness. Lastly, I will present existing definitions of a Brain-Computer Interface and try to explain the motivation behind the development of the Berlin Brain-Computer Interface (BBCI) presented in this work.

1. How this Thesis is Organized

This thesis is divided into five major parts; each chapter is self-contained and can be read separately or in random order. In the beginning will give an overview of the state-of-the-art of this field, honoring the pioneer research groups that have been developing Brain-Computer Interfaces for several decades. I will then present some of the most prominent systems and research groups. Each of these BCI-systems has its drawbacks; I will highlight these and compare them with the BBCI system proposed here. Readers already familiar with these topics may skip this part.

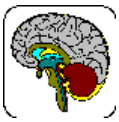
The guiding idea of every Brain-Computer Interface system is based on human neurophysiology; I will give an overview of this branch of biology in Chapter II, highlighting in particular certain pertinent “built-in” functions of the brain. A discussion on the influence of “real-time” and online bio-feedback on the behavior of the test person will conclude Chapter II. Please note that Chapter II does not claim to be complete nor detailed enough for deeper neurophysiological research; it serves rather as a description of how Brain-Computer Interfaces, in particular the BBCI system, operates. Readers familiar with basic principles of human neurophysiology may skip Chapter II. Conversely, readers who do not have basic knowledge of human neurophysiology are strongly advised to read Chapter II carefully before continuing with Chapter III.

Chapter III specifies and elaborates on the requirements and objectives of the Berlin Brain-Computer Interface and its hardware basis. This includes all sub-processes, listed in the sequential order of the usual data flow, in which they are carried out during various experimental setups. This then brings us to the core of this work: the bio-feedback modules. Section 7 of Chapter III will document several bio-feedback modules, including detailed descriptions of a series of conducted experiments.

Chapter IV will focus on my main contribution to this thesis, the state-of-the-art software engineering, object-oriented design and programming (OOD/OOP) approaches applied for the design and development of the BBCI prototype and several feedback modules. This includes, moving from the general to the specific, the design and specifications of the implemented system prototype. I will here present abstracts of data structures and the class hierarchy of the main part of the prototype, followed by the documentation of the major parts from the software engineering viewpoint. The chapter concludes with a documentation of the generic data formats. Readers not interested in implementation details may skip this chapter entirely.

In Chapter V, the last chapter, I will present perspectives and improvement propositions for the BBCI. Concluding the thesis is a visionary outlook of the future disposition and significance of Brain-Computer Interfaces in our modern information and computer society.

Some paragraphs of this thesis are highly technical. While of interest to technically-minded readers, these paragraphs are not central for understanding the thesis as a whole. These paragraphs will be designated with the “technical” symbol as illustrated here.



Some paragraphs provide or assume some knowledge of human neurophysiology; these may be skipped by readers who are not interested in these topics. Again, readers will still be able to follow the overall thread of the thesis. These specialized paragraphs are included in the thesis to provide completeness for the subset of readers that is interested in neurophysiological details. Please note that these paragraphs are kept as short and as simple as possible, omitting the gallimaufry of medical language. I will mark these paragraphs with the “brain” icon, as illustrated here.

The BBCI project is highly interdisciplinary and involves many researchers. I could therefore not even try to aim for completeness without mentioning the work done by my many colleagues. I often refer to work done by other team members – work that provides the basis for my own research which I then describe later. I will mark work done by other BBCI team members with the “team” icon, as illustrated here.

2. State-of-the-art in BCI

This section presents various techniques for “reading” information from the brain, followed by a summary of major pioneer research groups that have been developing Brain-Computer Interfaces for decades. The section’s subdivisions highlight respective paradigms that current BCIs incorporate. Finally, I will present an overview of a general-purpose BCI system, the BCI 2000. The BCI 2000, designed in an object-oriented and distributed manner similar to the

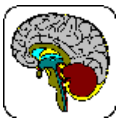
BBCI, is able to handle a variety of neurophysiological paradigms and can be adapted to control different bio-feedback applications.

2.1 Acquisition technology

All currently available acquisition technologies can be divided into two groups: invasive and non-invasive methods. The latter methods allow acquiring data without surgical interventions such that they can be applied to human test persons in ways that do not harm their health. These methods acquire data through electrodes placed on the surface of the head. That data is generated by the activity of large populations of neurons of the brain's cortex below the scull bone; as a result of the low conductivity properties of the scull bone, the generated data deliquesces at the inner side of the scull bone and is thus limited in its spatial resolution.

There are several non-invasive methods for monitoring brain activity; they include: Positron Emission Tomography (PET), functional Magnetic Resonance Imaging (fMRI), Magnetoencephalography (MEG) and Electroencephalography (EEG), all of which have their advantages and shortcomings.

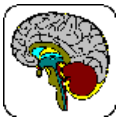
2.1.a Positron Emission Tomography (PET)



Positron Emission Tomography, or PET, is a procedure that allows a physician to examine the heart, brain and other organs. PET images show the chemical functioning of an organ or tissue, unlike x-rays; which show only body structure. PET imaging can show the region of the brain that is causing a patient's seizures and is useful in evaluating degenerative brain diseases such as Alzheimer's, Huntington's and Parkinson's. PET is able to indicate brain regions that are "active", i.e. regions that consume glucose or saccharose and that produce energy in exchange. However, it remains unclear to what extent positron irradiance is harmful to the human health. Moreover, the equipment and supplies are very costly for an everyday practice.

[`http://www.biomed.org/pet.html`]

2.1.b Near-infrared Spectroscopy (NIRS)



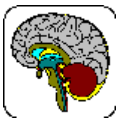
All of us are exposed to optical (i.e. visible and near-infrared) radiation from the sun and other sources throughout our lives. Assuming that our eyes are protected from excessive intensity and our skin from the ultraviolet content of the sunlight, we accept this exposure in the knowledge that it is perfectly safe. Unlike x-rays, optical photons are insufficiently energetic to produce ionization, and unless light is concentrated to such a high degree that it burns the skin, optical radiation offers no significant hazard. The diagnostic potential of optical methods has been widely known since Jöbsis [Jöbsis, 1977] first demonstrated that transmittance measurements of near-infrared (NIR) radiation could be used to monitor the degree of oxygenation of certain metabolites. This led to the development and increasingly widespread use of clinical near-infrared spectroscopy (NIRS), which offers a safe, non-invasive means of monitoring cerebral function at the bedside without the use of radioisotopes or other contrast agents [Cope and Deply, 1988].

Human tissues contain a variety of substances whose absorption spectra at NIR wavelengths are well defined and which are present in sufficient quantities to contribute significant attenuation to measurements of transmitted light. The concentration of some absorbers, such as water, melanin, and bilirubin, remain virtually constant with time. However, some absorbing compounds, such as oxygenated hemoglobin (HbO_2), deoxyhemoglobin (Hb) and oxidized cytochrome oxidize (CtO_x), have concentrations in tissue which are strongly linked to tissue oxygenation and metabolism. Increasingly dominant absorption by water at longer wavelengths

limits spectroscopic studies to less than about 1000 nm. The lower limit on wavelength is dictated by the overwhelming absorption of Hb below about 650 nm. However, within the 650-1000 nm window, it is possible with sensitive instrumentation to detect light that has traversed up to 8 cm of tissue.

[http://www.medphys.ucl.ac.uk/research/borl/research/NIR_topics/nirs.htm]

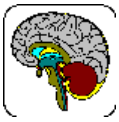
2.1.c Magnetic Resonance Imaging (MRI)



Magnetic resonance imaging, or MRI, is an imaging technique used primarily in medical settings to produce high quality images of the inside of the human body or brain. MRI applies nuclear magnetic resonance (NMR), a spectroscopic technique used by scientists to obtain microscopic chemical and physical information about molecules. The human body consists primarily of fat and water, both of which contain many hydrogen atoms; as a result, the human body consists in great part (approximately 63%) of hydrogen atoms. These atoms are dipoles and can be thought of as a small magnetic field; their nuclei will produce an NMR signal. MRI of the brain uses a magnetic field, radio waves and a computer to create detailed image slices of an area. MRI technology allows physicians to evaluate different types of tissue as well as distinguish healthy tissue from diseased tissue. Although MRI is able to provide high spatial resolution images, its temporal resolution is quite poor.

[http://www.cis.rit.edu/htbooks/mri]

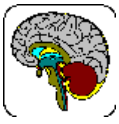
2.1.d Magnetoencephalography (MEG)



The measurement of magnetic fields generated by electric currents in the brain, close to the surface of the head, allows localizing the origin of electric currents and may be used to map cortical brain function. MEG is based on the principle that all electric currents generate magnetic fields. The main source of the extracranial magnetic fields detected with MEG instruments originate mainly from a current flow in the long apical dendrites of the cortical pyramidal cells. A distal excitatory synapse will induce a dipolar dendritic current towards the soma of the pyramidal cell, meaning that the electricity flows in one direction along the entire length of the dendrite, which can therefore be considered an electric dipole. Pyramidal neurons constitute nearly 70% of neocortical neurons. Their cells, with their long apical dendrites, are oriented perpendicular to the brain cortex. There are more than 100.000 of these cells per square millimeter. MEG provides millisecond temporal resolution and several millimeters spatial resolution of brain function; however, it gives no detailed anatomical information. Therefore it is often combined with MRI. The recorded bio-magnetic signals are very similar to EEG. Moreover, also similar to EEG, the signals may be either spontaneous or related to a stimulus, e.g. audiovisual, tactile, vibratory, or electric.

[http://www.amershamhealth.com/medcyclopaedia]

2.1.e Electroencephalography (EEG)



Electroencephalography (EEG) records the bioelectric activity of the brain by means of an array of up to 256 surface electrodes that, applied to the scalp, function based on volume conduction analogously to electrocardiograms. The voltage fluctuations registered by the EEG are summations of the continually ongoing electrical activity of large populations of cortical neurons. On the whole, an EEG record reflects the extracellular currents resulting from postsynaptic membrane depolarization and hyperpolarization of cortical pyramidal neurons. Their anatomical position (normal orientation to cortical surface, parallel array of dendritic tree, laminar distribution) results in the generation of relatively large electrical fields or dipoles.

EEG consists of the continuous recording of rhythmic positive/negative voltage fluctuations in the microvolt range (1-200 μV). It is state-dependent and reflects ongoing brain activity, thus, activity that may change due to age, arousal level, sleep stage, or cerebral dysfunction. It is mainly applied in the assessment of cerebral function rather than the detection of structural abnormalities. As such, it is applied to assess epilepsy, states of altered consciousness, head trauma and coma, anoxia, intoxications, cerebral infections and sleep disorders. Powerful, computer-driven analyses of EEG patterns could also be applied in the field of schizophrenia research and personality studies. It is important to note that clinical EEG recordings provide no information as to personality type and do not "read one's thoughts".

More specifically, EEG merely yields data that is easily recorded with comparatively inexpensive equipment, is rather well-studied and provides high temporal resolution. Thus outperforming other techniques, it is an excellent candidate for BCI. The temporal and spatial resolutions of all the above-mentioned acquisition techniques are summarized in Table 1.

Table 1: Temporal and spatial resolutions of the above-mentioned acquisition techniques

Acquisition technique	Temporal resolution	Spatial resolution
PET	~ 30 sec	4-6 mm
NIRS	~ 10 ms	20 mm
MRI	3-5 sec	~ 1.5 mm
MEG	~ 1 ms	~ 3-10 mm
EEG	< 0.5 ms	5-10 mm

EEG-based BCI systems can be subdivided into several groups according to the electrophysiological signals and paradigms they use.

2.2 Visual Evoked Potentials (VEP)

Visual Evoked Potentials (VEP) constitutes a dependent BCI, i.e. they depend on oculomotor control of gaze direction, such that activity in the normal information pathways (e.g. peripheral nerves and muscles) is needed to generate the brain activity. VEPs evaluate the visual nervous system from the eye to the brain. Sutter [Sutter, 1992] described a Brain Response Interface (BRI) by applying it as a keyboard interface: volunteer test persons selected one symbol from a set of 64 that were arranged in an 8×8 matrix; focusing on the selected symbol, they were then able to type 10-12 words/min. Symbols changed their color or flashed with a certain frequency, which induced a distinct spatiotemporal pattern in the visual cortex of the user's brain. However, this method requires stable control over the oculomotor muscles that are needed for focusing on a letter.

A dependent BCI is essentially an alternative method for detecting messages carried out in the brain's normal output pathways. However, it does not give the user a new communication channel that is independent of conventional channels. Visual Evoked Potentials are used to evaluate optic neuritis, optic tumors, retinal disorders and demyelineating diseases such as multiple sclerosis.

2.3 P300-based BCIs

BCI systems are by definition independent if they do not rely on any muscular activity, if the message is not carried by peripheral nerves and muscles, and, furthermore, if activation in these pathways is not required to generate the activity in the brain (as measured by an EEG) that carries the message. Independent BCIs attract more theoretical and practical interest than dependent BCIs because, offering the brain completely new output pathways, they are likely to be more useful for people with most severe neuromuscular disabilities.

For example, a user of a P300-based BCI may anticipate that a rare or infrequent auditory, visual, or somatosensory stimulus occur in the midst of standard or routine stimuli; at about 300 ms (P300) after the appearance of the awaited stimulus; a positive peak will occur over the parietal cortex. Donchin and Smith [Donchin and Smith, 1970] presented a P300-based BCI used for typing about 5 letters (or about one word) per minute. While the user faces a 6×6 matrix of symbols, its rows or columns flash alternatively every 125 ms in random order. The user then selects a symbol by counting how many times the row or column containing the desired selection flashes. Speaking in general terms, the user should pay attention to flash events of the desired symbol. In patients with visual impairments, auditory or tactile stimuli might be used [Glover et al., 1986].

P300-based BCIs have the clear advantage that they require no preliminary user training, as P300 is one of the brain's "built-in" functions. However, such techniques remained limited to letter or symbol selection paradigms, similar to the one described in the previous subsection. Moreover, long-term studies by Ravden and Polich [Ravden and Polich, 1999] have shown that P300 might habituate, with the effect that BCI performance decreases. An online adaptation of this learning machine will therefore likely become important for this kind of BCI.

2.4 Oscillatory features

Physiologically meaningful signal features can be extracted from various frequency bands of recorded EEG. For example, primary sensory or motor cortical areas often display 8-13 Hz activity (called μ-rhythm) when test persons are not engaged in processing sensory input or producing motor output (a similar fluctuation over the visual cortex is called α-rhythm). The μ-rhythms are usually associated with 18-25 Hz upper β-rhythms, largely known to be, to some extent, independent EEG features. Table 2 summarizes frequency bands (marginal frequency values are highly subject-specific) and the neurophysiological features they are assumed to encode.

Table 2: Frequency bands and associated brain functions

Band	Frequency [Hz]	Occur while / Indicate
δ	0.5 – 3.5	Movement preparation
θ	3.5 – 8	Memory
$\alpha\,(\mu)$	8 – 13	Relaxation (sensory idling)
β	13 – 25	Motor idling
γ	25 – 40	Feature binding

Pfurtscheller [Pfurtscheller, 1999] reports that μ- and/or β-rhythm amplitudes serve as effective input for a BCI. Movement preparation, followed by execution (or merely motor imagination) is usually accompanied by a power decrease in certain frequency bands; this power decrease is referred to as Event-Related Desynchronization (ERD). In contrast, their increase after a movement indicates relaxation and is due to the synchronization in firing rates of large populations of cortical neurons (ERS).

2.4.a BCIs based on Event-Related (De-) Synchronization (ERD/ERS) and on motor imagery

In Albany, New York, Jonathan Wolpaw directs the development of a BCI system that lets the user steer a cursor by oscillatory brain activity into one of two or four possible targets [Barkley, 1981]. People with or without motor disabilities learn to control their μ- or β-rhythm amplitude; an increase is associated with the cursor's movement to the target on top of the screen, while a decrease directs the cursor to the bottom. Users learn in 40-minute sessions that take place 2 to 3 times per week; most acquire significant control after 2 to 3 weeks of training. In the first training sessions, most subjects engage in some kind of motor imagery (e.g. imagining hand or feet movements, entire body activities, or even relaxed states). During further feedback sessions, these are then replaced by adapted strategies. Well-trained users are reported to achieve hit rates of over 90% in the two-target setup; however, each selection typically takes 4 to 5 seconds.

Gert Pfurtscheller's laboratory in Graz is developing a BCI system that is based on event-related modulations of the μ- and/or central β-rhythm of sensorimotor cortices. For a control paradigm, the focus is on motor preparation and imagination of simple actions, such as right-hand, left-hand or foot movements. Feature vectors, calculated from spontaneous EEG signals by adaptive auto-regressive modeling, are used to train a linear (LDA) or non-linear (LVQ) classifier or a neural network. In subsequent sessions the user is provided with online bio-feedback on the computer screen. This has allowed, for example, a quadriplegic test-person to simulate grasping movements of the hand by electrically stimulating its muscles. However, these movements could have been executed according to an externally synchronized communication protocol and sequence. In a ternary classification task, which was conducted in an offline manner by Peters et al. [Peters et al., 2001], users imagined the execution of a series of trials, each lasting over 8 seconds. About 90% of test persons could be able to seize significant control over the system, which was indicated by their classification accuracies of over 96%. Additionally, development of a remote control is addressed, allowing a BCI to operate in a user's home environment while the processing and classification is done in the laboratory.

2.5 Slow Cortical Potentials (SCP)

Slow Cortical Potentials (SCPs) are voltage shifts that are generated in the cortex and that last over 0.5-10 seconds. Slow negativation is usually associated with cortical activation (e.g. evoked by the implementation of a movement or by the accomplishment of a mental task), whereas positive shifts indicate cortical relaxation [Birbaumer, 1997]. Further studies showed that it is possible to learn SCP control. Consequently, it was used to control movements of an object on a computer screen in a BCI referred to as Thought Translation Device (TTD) [Birbaumer et al., 1999]. During the first 2 seconds, the system measures the user's initial voltage, followed by an action period of another 2 seconds where the user tries to increase or decrease her/his SCP by selecting one of the two targets; the result is then indicated on the computer screen as bio-feedback. After repeated training sessions that span several months,

patients achieve accuracies of over 75%. The patients are then switched to a letter support program, which allows a selection of up to 3 letters per minute.

A new predictive algorithm has been developed that performs a statistical analysis of the user's input. It calculates occurrence probabilities of single letters or entire words, e.g. more frequently used words obtain higher values. When the user types in the first letters of a word, the algorithm will then suggest an entire word, which is most probable according to the statistics. The algorithm is based on a lexicon, which is simultaneously updated to the user's vocabulary with each action. This increases the usability of such systems and exploits the comparatively low communication bit-rates. It is a conceivable future application to provide Internet access for disabled users [Birbaumer et al., 2000].

2.6 Invasive BCIs

Already in the 1960s, Evarts [Evarts, 1996] reported implanting microelectrodes into the cerebral cortices of awake animals; the microelectrodes then recorded the action potential of single neurons during the execution of movements. Further studies explored the capacity of animals to learn to control neural firing rates, such that the expectation that human could develop similar control capabilities for using them to communicate with their environment or to operate a neuro-prosthesis becomes evident.

Using information recorded invasively from an animal brain, Nicolelis and Chapin [Nicolelis and Chapin, 2002] discuss a BCI able to control a robot. Four arrays of fine microwires penetrated the animal's scull and connected to different regions inside the motor cortex. A robotic arm remotely connected over the Internet implemented roughly the same trajectory as the owl monkey grasping for food. Although this technique could be applied to humans in the near future, it still remains an unattractive method as it requires surgical intervention to implant the electrodes. Nevertheless, for patients with very severe neurophysiological disorders, it could well be an option to consider.

This invasive technology allows for the extraction of signals at extremely fine spatial and temporal resolutions; each microelectrode integrates firing rates of only a few dozen, or even single, neurons. Nevertheless, to make a BCI attractive for broad, everyday usage, it needs to be non-invasive, quickly mountable, in addition to leaving no bodily marks. In short, it should be as simple and harmless as putting on a baseball cap.

2.7 BCI 2000 – a general purpose BCI model

Because BCI development and application is a complex and multi-disciplinary endeavor, there is need for a system that readily adapts to a variety of brain signals, e.g. EEG signals or voltages or cortical neural activity, signal processing methods, translation algorithms and output modalities. Schalk et al. [Schalk et al., 2004] developed a generic and distributed system called BCI 2000 that can describe a wide variety of present-day and future BCI systems. It consists of four processes that interact through a defined interface: EEG acquisition, signal processing and translation, a user or feedback application, and an operator interface (see Figure 1). The modules communicate through a network protocol based on TCP/IP. The Source module passes current blocks of signal samples and event markers to the Signal Processing module. The Signal Processing converts this input into control signals and passes them and the state vector to the User Application. The User Application is controlled by these signals providing the user with the appropriate feedback and sends the state vector back to the Source. Because the state vector includes all relevant system events (e.g. device states, artifact

occurrence, etc.), its inclusion into the data file allows for full reconstruction of the experiment session and for comprehensive data analyses. The Operator module provides an interface to the experiment conductor to allow for system configuration and for the definition of onset and offset operation.

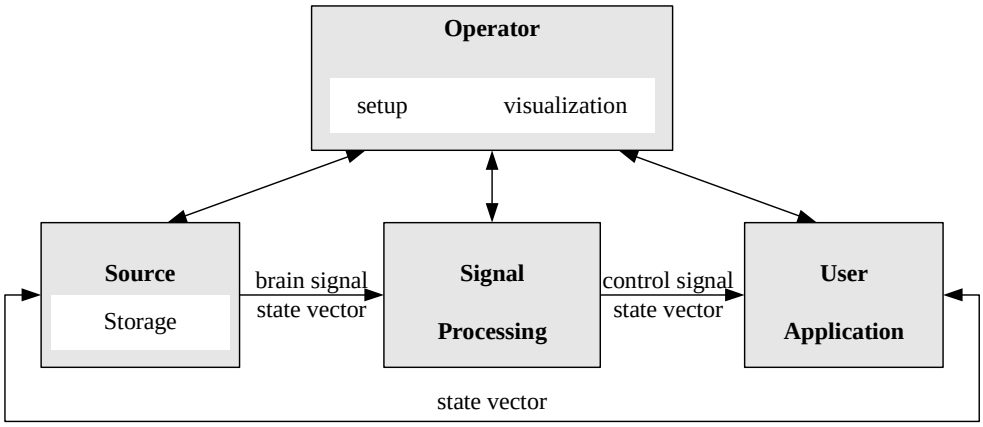


Figure 1: Design of the BCI 2000 system. Four modules are employed communicating via TCP/IP based protocols: Operator, Source, Signal Processing and User Application. The information (signals, parameters, event markers, i.e. the state vector) is communicated from Source to Signal Processing to User Application and back to Source. From [Schalk et al., 2004]

The BCI 2000 maximized the interchangeability and independence of the modules, since it is implemented with current techniques of Object-Oriented Software Design (OOSD). The communication between modules uses a generic protocol that can transmit all information required for operation. None of the four modules place any constraints on the number of channels or the sampling rate, the number of system parameters or event markers, the complexity of the signal processing, the timing of operation or the number of signals that control the output device. Thus, these factors are limited only by the capacities of the hardware used.

The first tests of the BCI 2000 were studies in which users could control cursor movement with one or a combination of several different EEG features.

3. Requirements and Drawbacks of Current BCIs

In the seven decades since Berger's original publication [Berger, 1929], electroencephalograms (EEGs) were used mainly to evaluate neurological disorders and to investigate brain function. In addition, researchers have been speculating that EEGs could be applied to decipher thoughts or intentions. If applied for this function, a person will be able to control devices directly by her/his brain activity, bypassing the normal channels of peripheral nerves and muscles. However, this speculation became technologically feasible only in the past decade, as the deciphering of thoughts or intentions required that a vast amount of data be analyzed in a limited time. A surge then ensued, promoted also by the rapid development in computer

hardware and software that enabled the distribution of complex system tasks among different computers that communicate with each other and that process acquired data in a parallel and real-time manner.

3.1 BCIs rely on brain activity

Recent research in digital signal processing (DSP) and data analysis offer the possibility to develop intelligent and automatically adapting systems. These systems do not rely on prior knowledge about the user; instead, they begin to setup the user model with the first contact with the user, adjusting it interactively during further online recording sessions. Obviously, the interaction of a test person and a computer application will induce certain spatio-temporal patterns in the cortices of the person's brain. In this sense, processing visual or auditory information provided by the monitor screen or loudspeakers produces specific EEG patterns in the primary visual or auditory somatosensory cortices. These patterns can be easily observed through EEG that is recorded over the respective sensory cortex and consequently classified in an automatic manner. Further cognitive processing of this information is then widely distributed across the cortex. Disentangling these contributions coming from different cortical processing modules poses a difficult challenge for non-invasive recording techniques. The intentions to control the system by performing motor (i.e. muscle) activity originate inside the brain's high-level decision centers – a core that cannot be reached by EEG. They then work their way through to the primary motor cortices such that rising neural activity can be recorded with EEG surface electrodes placed over the respective brain regions. Fortunately, these surface regions have a consistent somatotopic arrangement that represents the body in an orderly topography. This topography is often referred to as "Homunculus" and can be visualized as in Figure 2.

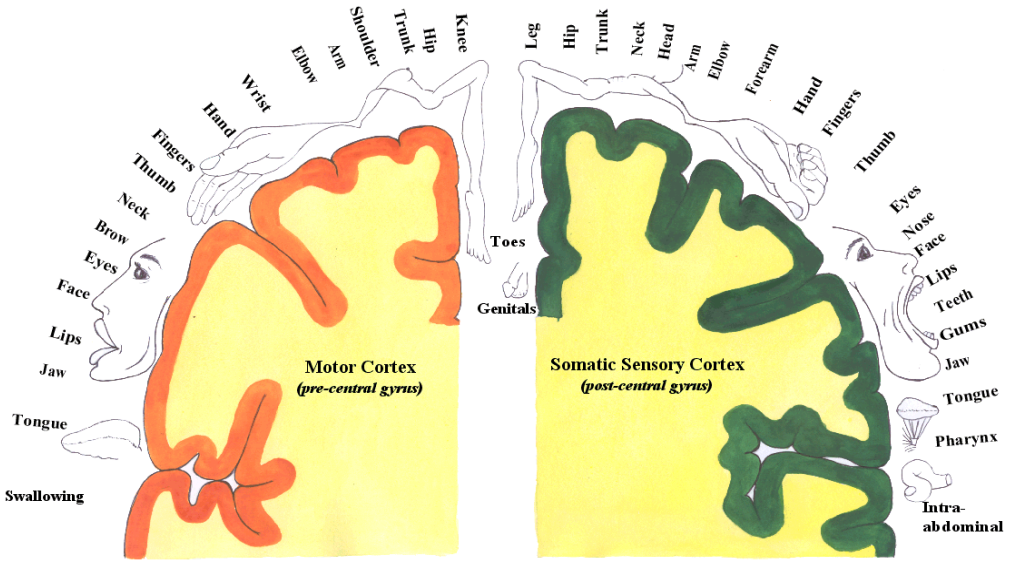


Figure 2: "Homunculus". A schematic distribution of human body parts according to the brain regions, which are responsible for their control (motor cortex on the left) or process information from them (somatic sensory cortex on the right).

The distinction of nearby located cortical areas still remains a difficult problem due to, for one, physical limits in the spatial resolution of surface EEG, and secondly, because each single electrode acquires superposed data from within a certain radius in which many different signals interfere.

3.2 The role of Human-Computer Interaction for BCI

Currently, modern Human-Computer Interaction (HCI) and multimedia technologies address only a subset of the I/O channels humans use to interact with computer applications or devices. Those require mainly motor (keyboard, joystick, pedal, etc.), visual (graphics, animation, etc.) and acoustic (sound, music, speech synthesis and analysis) senses. Recent research also tries to incorporate olfaction [Harel et al., 2003], tactile sensation [MacIntyre and Feiner, 1986], [Hardwick et al., 1996], interpretation of facial emotions providing additional information during communication [Pantic and Rothkrantz, 2000] and recognition of gestures [Pentland, 1995], [Quek et al., 2002]. Since all these information streams pass through their own interface (e.g. hands or skin, muscles, eyes, ears or nose) yet indirectly converge or emerge in the brain, the investigation of a direct communication channel between the application and the human brain should be of high interest to HCI and multimedia researchers [Ebrahimi et al., 2003].

Furthermore, Steriadis and Constantinou [Steriadis and Constantinou, 2003] state that development of human-computer interfaces for people with severe disabilities (e.g. quadriplegic due to amyotrophic lateral sclerosis (ALS) or spinal cord injury, or brainstem stroke patients) is highly important as it integrates them into our present-day information society. Their normal communication paths (e.g. peripheral nerves and muscles) required for interacting with computers or other devices having been damaged; however, information on intention of movement execution can be extracted from their last remained intact communication stage. Green et al. [Green et al., 1999] have shown that motor and sensory cortices of patients with amputated extremities (e.g. arms or legs) remain intact and produce normal spatio-temporal patterns similar to those of healthy people in regard to their intentions to move the absent part of the body. This implies that a technique for recognizing and deciphering those patterns and translating them into device control commands might serve as the basis for a wide variety of applications in the field of HCI. Ultimately, it will provide handicapped people with the ability to communicate with their environment or to control various devices.

A special role must be assigned to the instinctiveness of bio-feedback. Ramachandran et al. report in their work [Ramachandran et al., 1999] that after a lateral hemisphere stroke patients display an indifference to objects and events in the contralateral side of the world (neglect). Looking into a mirror while imagining moving the absent arm – a reflection of the other intact arm – helps to alleviate the phantom pain and to accelerate the recovery from neglect. Producing natural feedback on a computer screen with actions that are correlated to the patient's intentions might have similar positive consequences for the convalescence.

3.3 BCI for monitoring user states

However, the field of Brain-Computer Interfacing will in the future reach well beyond clinical and rehabilitative applications. Integrated into the headsets of pilots, train drivers or even astronauts, it may be able to provide users with an additional virtual hand for switching

various auxiliary devices on or off. For time-critical operations, such as emergency breaking of fast trains, these systems may extract the driver's intention a trickle of a second before the actual execution. Important, life-saving preparative operations (e.g. gritting sand on the rails) can then be commanded immediately.

Finally, the vast consumer community of gaming applications may soon be able to experience a completely new sentiment of incorporating own "thoughts" for virtual controlling of objects in games. A variety of, to date incredible, gaming scenarios will be developed in which individual users can train their own brain features or in which several BCI users can put their "brain-forces" to the test in round-robin-like games with competitive scenarios. These kinds of applications have already been used for therapeutic purposes. For example, children suffering from Attention Deficit Disorder (hyperactivity) have been playing a virtual version of the well-known "tug-of-war" game by training of relaxation mechanisms, which in turn increased their idling α- and β-rhythms [Barkley, 1981]. This method has also been widely applied since the late 1970s to treat stress and migraines and to help manage pain. Voelker et al. [Voelker et al., 1989], for their part, applied the method on students whose verbal and performance scores of the WAIS IQ test differed by more than 10 points. Training the less-performing cortical hemisphere of these students improved their performance significantly. Students with low verbal scores who were given stimulation with primarily β-frequencies over the left hemisphere, showed significantly improved scores after only several 30-minute training sessions.

In conclusion, Brain-Computer Interface systems can be effectively employed in clinical rehabilitation and convalescent therapies, as well as an assistance tool in time-critical applications, in innovative human-computer interfaces, in gaming scenarios for entertainment, or for training brain efficiency.

3.4 Requirements for developing a BCI system

A BCI system must be designed such that it can be comfortably carried out without any disadvantages for a person's health. Some further most important features for such a system include:

- For a BCI-system to be attractive for an everyday user it must be non-invasive, i.e. it must acquire information by placing surface electrodes on the scalp. Epidural and subdural techniques may be appropriate for most severely disabled patients; however, they should not be imposed on ordinary users.

- It should be quickly and easily mountable and removable – like taking a baseball cap on and off. Current technology still requires so-called "wet" electrodes to be filled with electrolyte gel before the experiment start; this is time-consuming in addition to making the procedure slightly uncomfortable and lasting.

- Functioning in real-time and online is important since bio-feedbacks make sense only when correlated with the user's actual actions. A system must thus be able to process the acquired data sufficiently fast. It has been shown in several feedback studies that delayed feedback or one, which is not correlated with the user's intentions leads to the deterioration of the performance (see Chapter II, Section 2 for detailed explanation).

- A BCI system should rely on independent brain features to provide even a healthy user with an additional information channel that can carry control messages independently from other existing communication channels, whether undamaged or damaged, active or resting. BCI systems based on brain features dependent to other communication paths provide its user with an appealing feeling of control, but not with an additional

communication channel. However, severely disabled patients are not able to receive any advantage from this kind of BCIs at all.

- It must require no or only minimal user training. Therefore it must rely on the human brain's "built-in" functions such that any person can use it after only a brief instruction by an advisor regarding the control strategy. This means that the user does not have to learn to produce certain spatio-temporal patterns in her/his brain's cortex, a process required by many of today's BCI-systems and which can last over several months.

- It must incorporate inexpensive and feasible technology, such as electroencephalography (EEG), in order to compete with other acquisition methods and to ultimately become less expensive for the end-user. My belief is that cheap BCI modules will be integrated into gaming consoles in the nearest future. Furthermore, BCI technology can be successfully employed for user monitoring purposes, such as tiredness of truck drivers; however, for being installed into trucks it must be of a reasonable cost-performance ratio.

- It must be reliable with respect to decision errors. This means that the computer system must be flexible enough to detect the spatio-temporal patterns a user generates when performing a task, and to automatically update its knowledge such as to better suit the user's brain properties. For this purpose, an error detection mechanism can be incorporated which monitors bio-feedback sessions and controls the knowledge update each time a feedback error takes place. User performance is reported to vary during long experiment sessions, as well as users are expected to change their control strategies if becoming tired. For this reason continuous re-learning strategies can be employed to adjust the present user model.

- Last but not least, the "fun-factor" for healthy operators of these types of interfaces (e.g. dexterity games or other interactive applications) should not be underestimated. The large machinery of the Computer Gaming industry offers the opportunity for BCI systems of being used by a wide society of customers. This may result in fundamental funding of further BCI research and development.

4. Motivation for the Berlin-BCI

The first valuable definition of a Brain-Computer Interface was given as early as 1973 by J. J. Vidal. In his pioneering work [Vidal, 1973], a BCI-system was defined as any computer-based system that monitors human brain activity and that is able to decode that activity in a manner suitable for the control of any application. Since that time – more than three decades of intensive research – the definition of a BCI-system changed repeatedly according to the technical potentials and possibilities provided by the rapid development in computer science and newly consolidated neurophysiological findings. A more recent review on Brain-Computer Interfaces defines a BCI as a system for controlling a device (e.g. a computer, wheelchair or neuroprosthesis) by human intentions, which does not depend on the brain's normal output pathways or peripheral nerves and muscles [Wolpaw et al., 2002].

There is a strong incentive to develop a BCI system, that (*i)* matches the latest definition given by the research community, (*ii)* overcomes the limitations of the other, above-mentioned current BCI-systems and (*iii)* meets the challenging requirements set up to warrant its attractiveness for the everyday user.

The acknowledgment that BCI system development, despite its impressive expansion in recent years, is largely deprived of proper real-time support was an important motivational factor for my scientific investigation that culminated in this dissertation. Another determinant factor for this dissertation was the applicability of Brain-Computer Interfaces to significantly

improve the quality of life for a large number of patients suffering from severe impairments, including amyotrophic lateral sclerosis (ALS) or spinal cord injuries. With BCI systems, these patients will in future be able to control their wheelchairs directly through their thoughts, write texts in order to communicate with their environment, or even control computer applications, such as Web browsers or computer games. It is important to provide these capabilities to all people irrespective of their impairments for participating in our current day computer and Internet society.

4.1 Aims of this work

Many researchers with different background knowledge work on the development of the BBCI system. In recent years computer science emerged as a meta-discipline that combines these several disciplines to an union aiming at producing valuable results; it helps instantiating scientific investigation and implementing research ideas, e.g. resulting in an end-user product, a feasibility study or a computer based simulation. This dissertation ranges in the computer science domain; thus my job was to acquire the results of earlier research and to unify it in an inventive way. The purpose of this endeavor is to bring the BCI technology and in particular the applications based on it nearer to the community of potential users.

The goals of this doctoral thesis, i.e. my own part within the scope of the BBCI project, can be summarized as follows:

- To investigating the feasibility of real-time approaches for being applied to the BCI model. This aims at selecting of the most adequate real-time approach for the design of an online architecture for the BBCI system.
- To design, develop and implement the prototype – a general-purpose BCI system – that can be used to further research powerful digital signal processing (DSP) procedures and statistical classification methods. These can, in turn, be incorporated into future BCI applications.
- To investigate the wide range of various feedback applications with respect to their suitability for being controlled by a BCI system. It is of major importance to setup the constraints for the design and development of feedback applications, which should be controlled by a BCI system.
- To design and validate various control strategies for feedback applications. Most current-day applications are designed for the control by common interface devices. A BCI provides the user with a different information type, such that appropriate control strategies should be investigated for transmitting this information to an application or a device.
- To analyze the user's behavior during the BBCI experiments. Most important is the confrontation of the user with the real-time and online feedback animation that provides the user with instant control of its objects. In particular, the presentation of the user's intention a trickle of a second before the intrinsic execution, i.e. the confrontation of the user with its own future act is worth to be investigated.

Chapter II —

HUMAN NEUROPHYSIOLOGY —

AN OVERVIEW

Der Mensch ist das Modell der Welt
Leonardo da Vinci (1452 - 1519 A.D.)

Before delving into the complexities of BCI research, we need to understand the main processes that take place in the human brain. It is for this reason that I included this chapter which aims to familiarize the reader with the most prominent features of human neurophysiology. It will introduce basic neurophysiological terminology, which will then be used in further explanations; however, it does not claim to be complete. The chapter is divided in two parts. The first describes a variety of brain functions and paradigms that can be observed from EEG data. This data is classified as either phase-locked (i.e. "plain") and phase-unlocked (i.e. oscillatory). The second part discusses the main information pathways in humans and the problems and conditions related to their usage. Reading this chapter should, furthermore, help to better understand the issues described in Section 2 of the previous chapter.

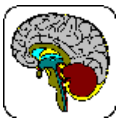
1. "Built-In" Functions of the Brain

Granted, this section's title is highly debatable as most or to some extent all functions of the human brain are "built-in". However, I will focus on those functions that are most promi-

nently observable through EEG signals; these do not require long training sessions yet are present in “normally behaving” subjects.

Basically, all signals can be divided in two groups depending on the acquisition domain under consideration. For example, slow amplitude shifts, among other potentials, are visible in raw EEG data within the time domain. Oscillatory features for their part, i.e. the amplitude within certain frequency regions (e.g. band power of oscillations), can be easily extracted from the frequency domain. These two signal properties are known to be induced from different mechanisms and are therefore believed to be independent. Section 1.3 will present special features of both paradigms that encourage the interaction of the user’s act of imagining with the online bio-feedback. In section 2 the emphasis is then given to the influence of the real-time interaction with the feedback application on the user’s behavior.

1.1 Slow Cortical Potentials (SCP)



Electroencephalography is a method to investigate human brain functions including ways of determining the reaction of the brain to a variety of stimuli. The great majority of these reactions remain hardly detectable in the EEG and only very few are easily visualized. The research field dedicated to the detection, quantification and physiological analysis of those slight EEG changes, each of which is related to a particular event, has been of growing interest in recent years. These EEG changes are referred to as “Event-Related Potentials” (ERPs) [Lopes da Silva, 1999].

ERPs are usually defined in the time-domain as the brain electrical activity that is triggered by the occurrence of particular events or stimuli. The basic analytical challenge is detecting ERP activity within the often much larger ongoing or background activity, i.e. the activity not related to the stimulus [Dawson, 1951]. Dawson [Dawson, 1954] attempted to solve this problem by the photographic superposition of a number of time-locked responses. This method has the advantage of enhancing response in contrast to the ongoing activity while, at the same time, providing an indication of response variability. It is, however, difficult to quantify the results in this way.

According to the most widely accepted model, ERPs are signals generated by neural populations that are activated time-locked to the stimulus; these signals are summed to the ongoing EEG activity. From another point of view, ERPs are assumed to result from the reorganization of part of the ongoing activity (see Section 1.2 for a more detailed description); however, this viewpoint needs further investigations.

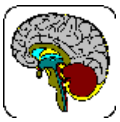
The analysis begins with two basic assumptions: that *(i)* the electrical response evoked from the brain is invariably delayed relative to the stimulus and *(ii)* the ongoing activity is a stationary noise, the samples of which may or may not be correlated. In this case, the ERP can be considered as a signal $s(t)$ corrupted by additive white noise $\eta(t)$. Consequently, the recorded signal $x(t)$ is given as a sum of two terms: $x(t) = s(t) + \eta(t)$, with the zero-mean stochastic noise. The expected value of the average of $x(t)$ over a number N of realizations, i.e. trials, is given as

$$\langle x(t)\rangle = \left\langle \frac{1}{N}\sum_{i=1}^{N} x_i(t)\right\rangle = \frac{1}{N}\sum_{i=1}^{N}\langle s(t)\rangle + \frac{1}{N}\sum_{i=1}^{N}\langle \eta(t)\rangle \underset{N\to\infty}{\to} s(t), \tag{1}$$

since $\langle \eta(t)\rangle = 0$. Therefore, the signal-to-noise-ratio (SNR) in terms of amplitude improves with the order of $O\left(\sqrt{N}\right)$. For a comprehensive study of ERP, multivariate analysis methods may be used [Streeter and Raviv, 1965], [John, et al., 1964], [Ruchkin et al., 1964] and [Donchin, 1966].

In detecting ERPs, the main concern is, of course, to increase the SNR so that the EEG background activity will not contaminate the ERP. The ratio between the variance of the averaged signal and the variance of the noise is $1/N$. This holds for the case where the noise samples are uncorrelated. Often, however, this is not the case, as for instance in the presence of a strong rhythmic background such as α-rhythm. In such a case, subsequent samples of $\eta(t)$ are not independent; their degree of dependency is given by the result of the autocorrelation function of the background activity. The detection of single-trial ERP is important for two reasons: (*i)* to remove the effect of latency variations between single-trial ERP, which leads to a deterioration of the ensemble average and (*ii)* to classify single-trial ERP [Lopes da Silva, 1999].

1.1.a P300 potentials



The most prominent specification of the ERP is its so-called P300 component, indicated by a strong positivation of cortical voltage at about 300 ms after the stimulus. Even though much progress has been made since the P300 potential was discovered (for extensive review see [Sutton et al., 1965], [Bashore and van der Molen, 1991] and many others), the primary question remains: What cognitive events are reflected by the P300 component? The answer is not yet clear, but current investigations of P300 are based on (*i)* neurophysiological investigations of the brain mechanisms that underlie its generation, (*ii)* evidence from experimental studies that manipulate psychological variables, and (*iii)* results obtained from neurophysiological reports that examine the associations between P300 and behavioral test data. Additional reviews and perspectives are available elsewhere [Donchin et al., 1986], [Johnson, 1988], [Oken, 1989], [Picton, 1992].

An important consideration of P300 generation stems from the observation that intrusive or novel stimuli (e.g. dog barking, abstract color forms) can produce an early positive-going component (the P3a), with a subsequent peak (the P3b) that is considered as the canonical P300 [Squires et al., 1975]. The P3a is generally larger in amplitude over the frontal and central electrode sites compared to the P3b and is thought to reflect an alerting process that originates in the frontal cortex [Courchesne et al., 1975]. However, recent evidence suggests that the P3a can be elicited with ordinary stimuli by using a three-stimulus oddball task. In the auditory version of this paradigm, simple tone pitches can be employed (e.g. low, medium and high tones) such that a rare target, a frequent standard, and an infrequent non-target are presented in a random order to the subject responding only to the target. When the discrimination between the target and standard stimulus is made difficult (e.g. by approaching their pitch frequencies), so that focused attention is engaged, the infrequent non-target stimulus will produce a P3a component that is larger in amplitude and earlier in latency than the P3b component produced by the target stimulus. Similar results have been obtained for typical auditory and visual stimuli [Comerchero and Polich, 1999]. These findings suggest that the P3a reflects the interruption of strong attentional engagement of the frontal lobe and is observable when infrequently presented stimuli interrupt primary task processing [Knight, 1996].

Figure 3 illustrates typical ERP curves, obtained from the classic oddball paradigm experiment (left column) where the user was visually provided with frequent non-target and rare target stimuli. The user was supposed to acknowledge the target stimuli only. In the novelty oddball paradigm experiments (right column), the user was additionally provided with an unexpected novel stimulus. A strong positivation (please note a flip of the ordinate axis) – the P300 peak – can be seen in all plots at about 330-350 ms after the stimulus presentation, with maximum positivation at centro-parietal electrodes.

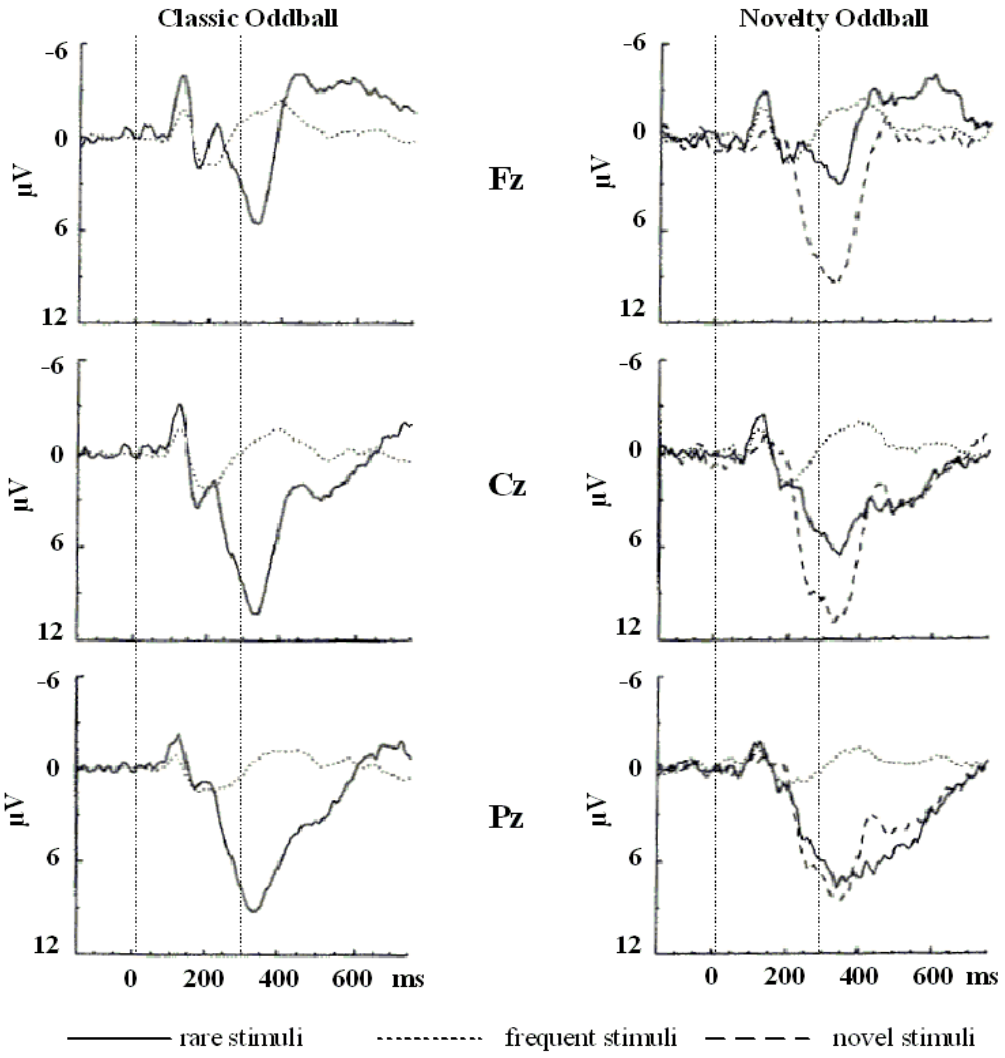
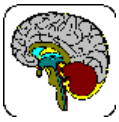


Figure 3: Typical ERP curves acquired from the classic oddball (left column) and novelty oddball (right column) paradigm experiments. Data is averaged with the stimulus (0 ms) as trigger and illustrats the P300 potential. From [Spencer et al., 1999]

1.1.b Error-Related Potentials (ERP)



Falkenstein et al. described the negative (*Ne*) and a later positive (*Pe*) deflection in the event-related brain potentials of incorrect choice reactions [Falkenstein et al., 1990] and [Falkenstein et al., 1991]. Originally, they assumed the *Ne* to represent a correlate of error detection, i.e. a mismatch signal when representations of the actual response and the required response are compared. This hypothesis was corroborated by the results of a variety of experiments and also by [Gehring et al., 1993], [Bernstein et al., 1995], and [Scheffers et al., 1996]. However, *Ne* was also observed after correct responses. Since the above-mentioned process is also required after correct responses, it is conceivable that *Ne* reflects this comparison process itself rather than its outcome. As for *Pe*, it is believed to be a further error-specific component, which is independent of the *Ne*, and is hence associated with an advanced stage of error proc-

essing or post-error processing. However, to specify the functional significance of *Pe*, further research is necessary [Falkenstein et al., 2000].

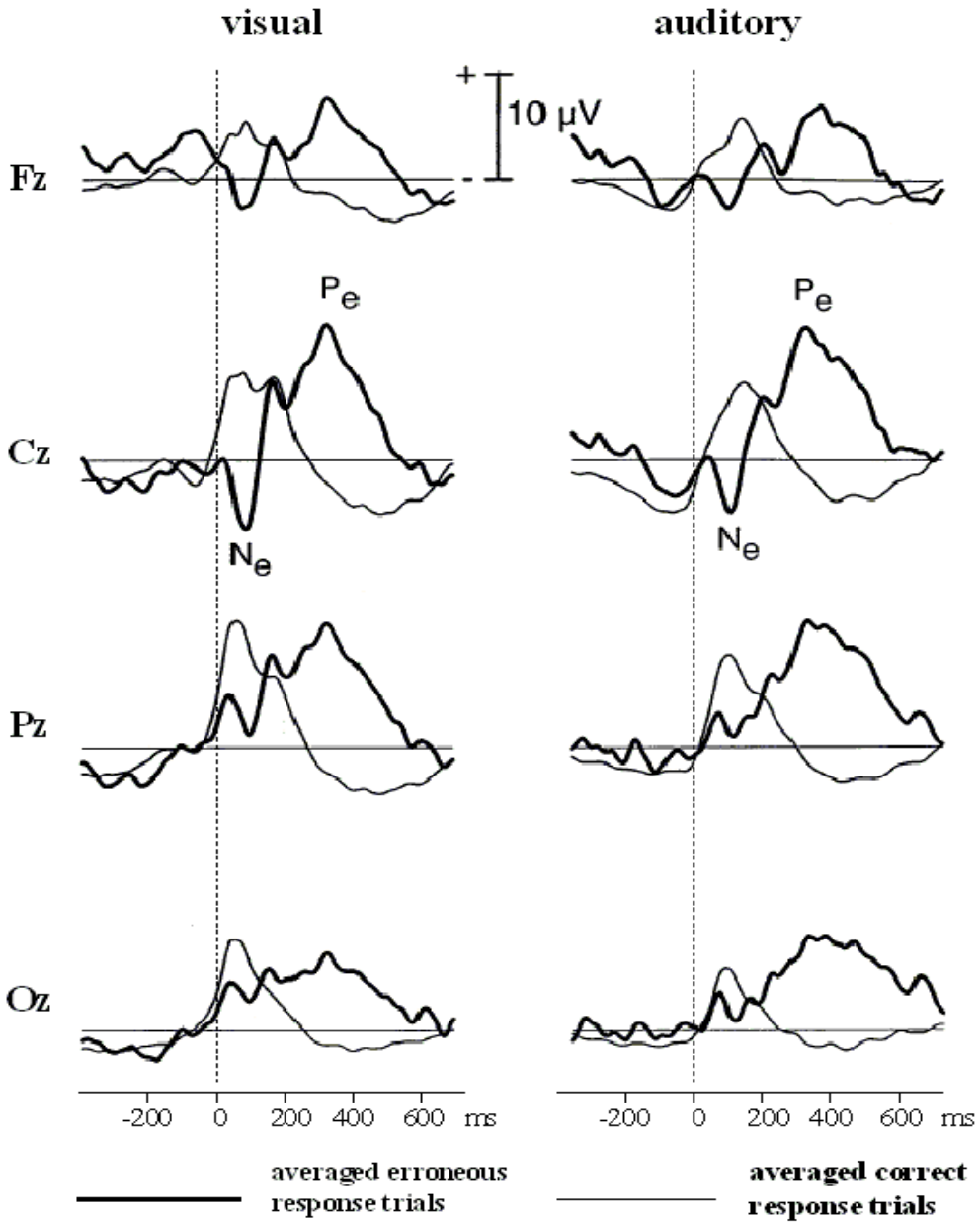


Figure 4: Typical ERP curves from experiments with visual and auditory stimuli selected from frontal, central, parietal and occipital electrode positions. Data is averaged with the response (0 ms) as trigger separately for mismatched and correct trials and indicates the error-related potentials. From [Falkenstein et al., 2000a]

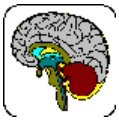
Figure 4 illustrates typical ERP curves which have been acquired in a 2-class-response experiments, where the user was displayed the letters ‘F’ or ‘J’ on a computer screen for the

visual case, or provided with two different types of tones via headphones for the auditory case. The user had to react as fast as possible by hitting a key on a computer keyboard with either the left or right-hand index finger, i.e. the letter keys ‘F’ or ‘J’. After performing several sessions, all trials were averaged in two groups, the correct user responses composing one group, and the incorrect user responses the other. To distinguish the groups, time windows were aligned at the response event marker that appeared at the time point of the key press.

Error negativity (*Ne*) is seen as a sharp negative deflection with a central maximum peak at about 80 ms after the incorrect key press. Error positivity (*Pe*) is seen as the late parietal positivation, with Cz peaking maximally at about 300 ms after the incorrect key press. On correct trials, a positive complex with Pz maximum is seen.

Regarding error positivity (*Pe*), Falkenstein et al. [Falkenstein et al., 2000a] performed a large variety of experiments with different paradigms and concluded that this component reflects additional processing after errors, which is functionally different from error detection or response checking (as probably reflected in the *Ne*). This additional processing is reflected in the P3-like wave, which is elicited by the error event. Such additional processes could also reflect the subjective or emotional assessment of errors; this means that each individual user accords their own significance to errors.

1.1.c Readiness Potentials (Bereitschaftspotential)



Ever since we have had evidence of the Supplementary Motor Area (SMA), which is associated with voluntary movements [Deeke and Kronhuber, 1978], different methods of functional brain imaging have shown that medial-wall motor areas and the primary motor cortex (MI) are activated when healthy subjects perform repetitive, voluntary finger movements. Further evidence comes from electrophysiology in non-human primates, showing that neuronal activity in the above-mentioned areas and the Lateral pre-Motor Cortex (LMC) precedes the initiation of self-initiated movements [Tanji, 1994], [Razzolatti et al., 1996]. The main argument that neuronal activities in the fronto-central mesial cortex are not epiphenomena comes from observations in patients with lesions of that part of the brain. In the acute state, lesions of the fronto-central mesial cortex cause akinesia, defined as a lack of internally mediated voluntary movements of the contra-lateral part of the body [Förster, 1936].

A main controversy is whether the medial-wall motor areas and the lateral motor areas form a functional network that together sub-serves the organization of a voluntary movement, or, whether there is a functional specificity and hierarchy. One hypothesis concerning that controversy is that the activity of the medial-wall motor areas precedes that of the primary motor cortex. This hypothesis implies that there is a temporal sequence in the brain processing that underlies the transduction of an intention to act into overt behavior [Deeke et al., 1976].

The brain potential that precedes the initiation of a voluntary movement is called readiness potential or Bereitschaftspotential (BP). It starts at about 1.5 s before movement onset in electrodes located over the medial-wall motor areas and is of negative polarity when using a far-away reference [Cui et al., 1999]. It has been suggested that the BP at its onset results from a radial current sink which is caused in the medial-wall motor areas [Deeke et al., 1976], [Lang et al., 1990]. EEG has the advantage of being sensitive not only to tangential currents, as is MEG, but also to radial currents. BP topography changes about 750 to 500 ms before movement onset, which is characterized by an increasing lateralization of the BP with larger amplitudes above the contralateral primary motor cortex (MI) as compared to amplitudes above the ipsilateral MI [Deeke et al., 1976].

Figure 5 shows typical BP curves, which were acquired from self-initiated movement experiments [Kukleta and Lamarche, 2001] where the user was instructed to perform brisk finger movements. All single-trials have been averaged at the user response (0 ms) as trigger.

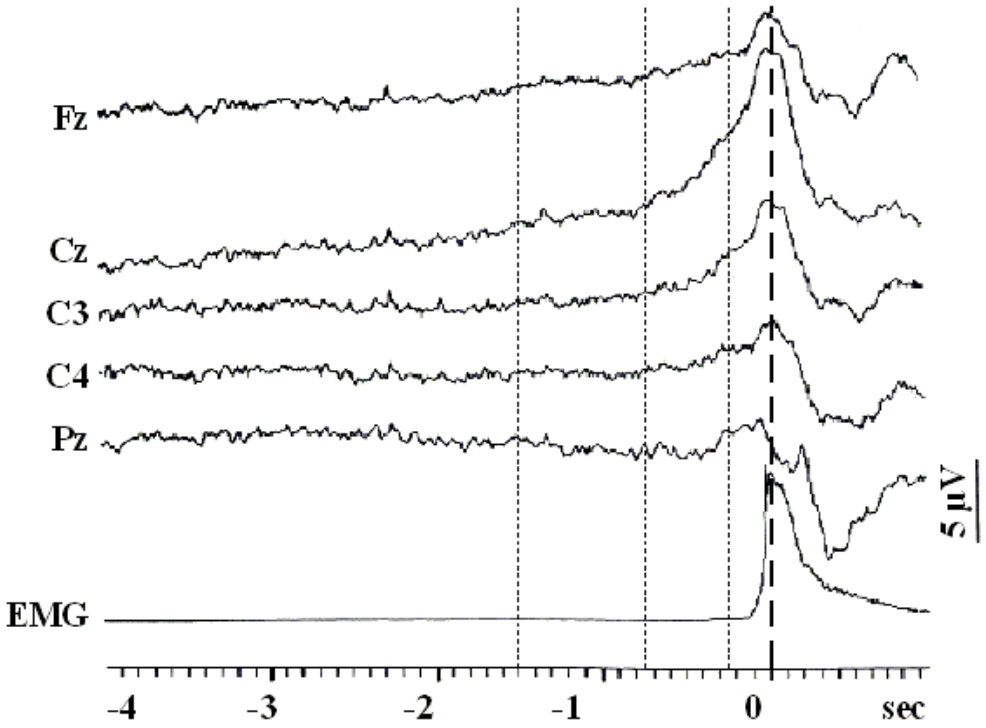
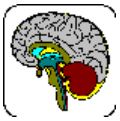


Figure 5: Typical ERP curves at positions of Fz, Cz, C3, C4 and Pz electrodes. Data is averaged at the user response (0 ms) as trigger and illustrates the readiness potentials. EMG is shown for control reasons. Dashed lines at -1.5, -0.75, -0.25 sec indicate changes in the variation of the BP slope. From [Kukleta and Lamarche, 2001]

1.2 Event-Related (De-)Synchronization (ERD/ERS)



An internally or externally paced event results in a change of the ongoing EEG in the form of an Event-related Desynchronization (ERD) or Event-related Synchronization (ERS). This response on the brain's neuronal structures is not phase-locked to the event and is highly frequency-band specific, whereby either the same or different locations on the scalp can display ERD and ERS simultaneously [Pfurtscheller and Lopes da Silva, 1999].

Several kinds of events, most notably sensory stimuli, can induce time-locked changes in the activity of neuronal populations; these changes are generally called event-related potentials (ERPs) (see Section 1.1.). In order to detect such ERPs, averaging techniques are commonly used. The basic assumption is that the evoked activity, or signal of interest, has a more or less fixed time-delay in regard to the stimulus, while the ongoing EEG activity behaves as additive noise. The averaging procedure will enhance the signal-to-noise ratio. However, this simple and widely used model is just an approximation of the real situation. Indeed, evoked potentials (EPs) are considered to result from a reorganization of the phases of the ongoing EEG signals [Sayers et al., 1974]. In addition it was also shown that visual stimuli can reduce the ongoing EEG amplitude [Vijn et al., 1991], thus invalidating the general model assuming that an ERP can be represented by a signal added to uncorrelated noise. Furthermore, it has been known

since Berger's findings [Berger, 1930] that certain events can block or desynchronize the ongoing alpha activity. These types of changes are time-locked to the event but not phase-locked, and thus cannot be extracted by a simple linear method, such as averaging, but may be detected by frequency analysis. This means that these event-related phenomena represent frequency-specific changes of the ongoing EEG activity and may consist, in general terms, of either decreases or increases of power in given frequency bands. This may be considered to be caused by a decrease or an increase in synchrony of the underlying neuronal populations, respectively. The former case is called Event-related Desynchronization or ERD [Pfurtscheller, 1977], [Pfurtscheller and Aranibar, 1977] and the latter is called Event-related Synchronization (ERS) [Pfurtscheller, 1992]. Of course, ERD and ERS phenomena are not only found with EEG but also with MEG recordings.

In general, the frequency of brain oscillations is negatively correlated with their amplitude, which means that the amplitude of fluctuations decreases with increasing frequency. For example, the Rolandic μ-rhythm with a frequency between 8 and 13 Hz has a larger amplitude than the central beta rhythm with frequencies around 20 Hz. The beta rhythm, in turn, has a larger amplitude than oscillations around 40 Hz. Because the amplitude of oscillations is proportional to the number of synchronously active neural elements [Elul, 1972], slowly oscillating cell assemblies comprise more neurons than fast oscillating cells [Singer, 1993]. This is valid not only when comparing oscillations around 10, 20 and 40 Hz, but also for components within the individual frequency bands.

One of the basic features of ERD/ERS measurements is that the EEG power within identified frequency bands is displayed (as a percentage) relative to the power of the same EEG derivations recorded during the reference or baseline period a few seconds prior to the event. Because event-related changes in ongoing EEG need time to develop and recover, especially when alpha band rhythms are involved, the interval between two consecutive events should last at least a few seconds. In the case of voluntary limb movement studies, an inter-event-interval of approximately 10 seconds is recommended; in the case of a fore-period reaction time task, the interval should be even longer. The classical method to compute the time course of ERD includes the following steps:

- Step 1: Band-pass filtering of all event-related trials;
- Step 2: Squaring of the amplitude samples to obtain power samples;
- Step 3: Averaging of power samples across all trials;
- Step 4: Averaging over time samples to smooth the data and reduce variability.

This procedure results in a time course of band power values, including also phase-locked and non phase-locked power changes. To distinguish the two types of power changes, the above-mentioned procedure should be complemented with another one where the step of squaring the filtered amplitudes is omitted; its steps are as follows:

- Step 1: Band-pass filtering of all event-related trials;
- Step 2: Calculation of the point-to-point inter-trial variance;
- Step 3: Averaging over time.

The difference between both ERD/ERS calculating procedures is described by Kalcher and Pfurtscheller [Kalcher and Pfurtscheller, 1995]. It was found that in the case of lower frequency components (lower α and θ bands) a phase-locked power increase due to the ERP can mask the non-phase-locked power decrease (ERD) when the classical band-power method is used.

To obtain percentage values for ERD/ERS, the power within the frequency band of interest in the period after the event is given by *A*, whereas that of the preceding baseline or refer-

ence period is given by *R*. ERD or ERS is defined as the percentage of power decrease or increase, respectively, according to the expression ERD% = $(A - R)/R \times 100\%$. For the display of the time course of ERD/ERS, a scale displaying either power changes with 0% in the reference period or relative power with 100% in the reference period is recommended. The total procedure of ERD/ERS calculation is displayed in Figure 6 from [Pfurtscheller and Lopes da Silva, 1999], with one example of dominant ERD in the α band on the left side and one example with dominant ERS in the β band on the right side. Further details can be found elsewhere [Pfurtscheller, 1999a].

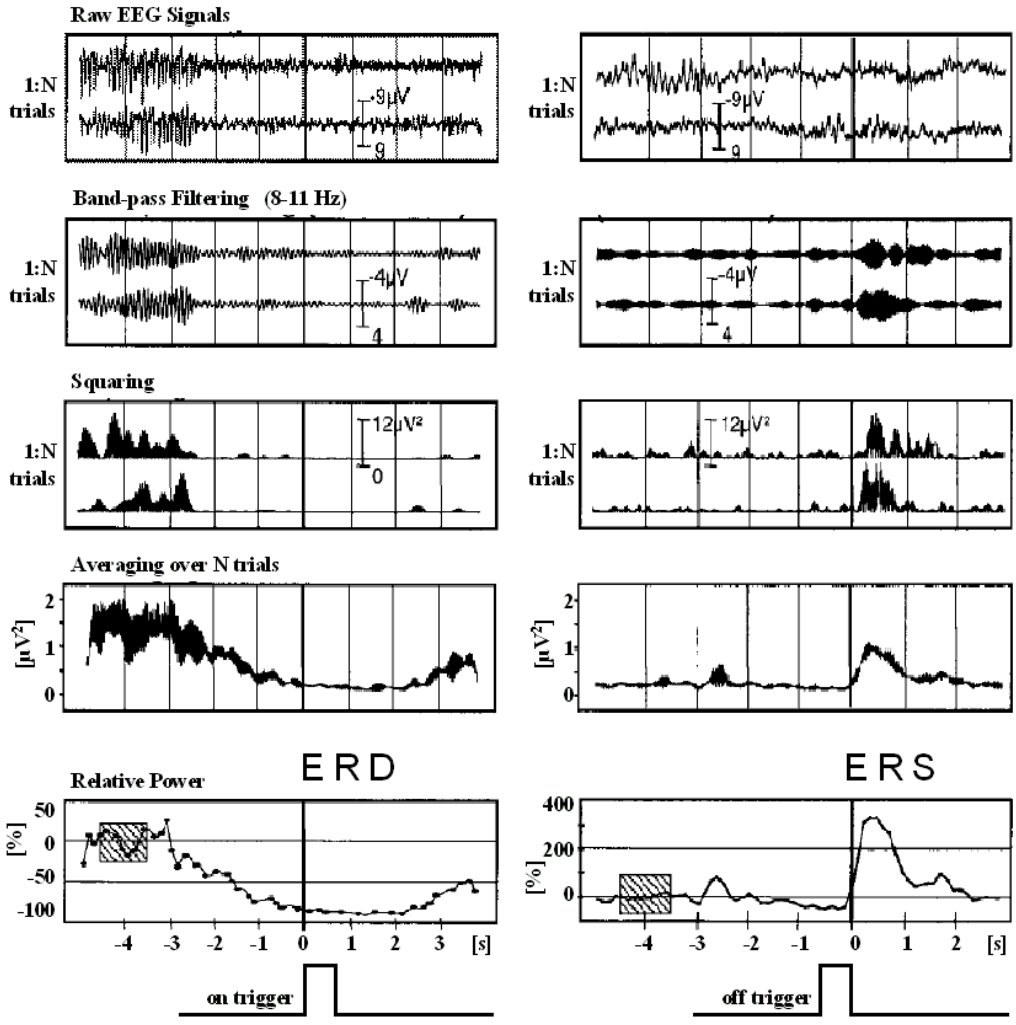
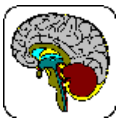


Figure 6: Principle processing scheme of ERD (left column) indicated by a decrease of power in the α band and ERS (right column) with an increase of power in the β band. From [Pfurtscheller and Lopes da Silva, 1999]

1.3 Imagery



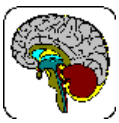
Internal Simulation of Movements (ISM) has gained increasing attention since the pioneering work of Decety et al. [Decety et al., 1988], [Decety, 1993]. Several previous experiments investigated neurophysiological processes underlying ISM [Ingvar and Philipson, 1977], [Goldenberg et al., 1986]. A theory was put forth, arguing that similar brain regions that are concerned with the actual execution of a movement are also activated during the mental ideation of that movement [Decety and Ingvar, 1990].

This hypothesis gained further evidence from neurophysiological observations of normal subjects [Decety et al., 1989] as well as of patients with lesions affecting different levels of the motor system [Dominey et al., 1995]. In these experiments, the performance times of the actual execution and the mental imagination for the same type of movement were very closely related. However, the studies yielded contradictory results as to differences in the topographical patterns of cerebral activation between mental ideation and the actual execution of a movement. This pertains especially to the question of a joint activation of the supplementary motor area (SMA) and the primary motor cortex (MI). Whereas some studies described a lack of activation of MI during ISM, it was recently shown with PET, DC-potentials, MEG, fMRI and transcranial magnetic stimulation, that MSI does imply an activation of MI [Höllinger et al., 1999].

Another topic in imagery research concerns hemispheric specialization for the different steps underlying the imagination of visual objects [Farah, 1984] and movements [Goldenberg et al., 1986]. In a previous study [Beisteiner et al., 1995] performance-related slow brain potentials were significantly larger in the left hemisphere compared with corresponding recordings from the right hemisphere. Moreover, left hemisphere dominance was significantly larger with internal simulations than with the execution of bimanual finger movements (yet not with movements of one hand separately). It was speculated that this difference might be due to larger demands on spatial coordination during ISM of bimanual than of unilateral movements [Höllinger et al., 1999].

The following subsections will briefly describe some of the most prominent types of imagery paradigms; this will include their underlying neurological processes as well as the electroencephalographic effect we expect there from.

1.3.a Motor imagery

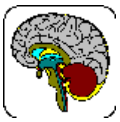


Motor imagery can be defined as the imagined rehearsal of a motor act without any overt movement. Internally we realize the ability to simulate a movement within its temporal and spatial sequencing and, by doing so we produce images of sensation which would arise during execution [Beisteiner et al., 1995]. A current point of interest is the functional similarity between imagined and executed movements. A close functional relationship has been suggested on the basis of several observations. Imagined and executed movements have similar durations and similar consequences on vegetative parameters. When subjects imagine performing a movement on their own accord, specific muscles corresponding to the simulated motor act are activated. Yet, while motor imagery is similar to the execution of a motor skill, the returned feedback results cannot be perceived and processed [Mendoza and Wichman, 1978].

Current discussions of the physiology and anatomical organization of motor imagery should not overlook past observations on the human EEG which frequently noted that the mere thought about a movement blocks the μ-rhythm [Chatrian et al., 1959]. Blocking even occurred when persons with amputated limbs imagined moving the phantom limb [Klass and Bickford, 1957]. The central μ-rhythm, which has already been described above, is blocked prior and during hand movements in a small and distinct scalp area located above the

hand area of the sensorimotor cortex [Pfurtscheller and Aranibar, 1978]. The μ-rhythm is clearly related to a strictly localized β activity over the human motor cortex

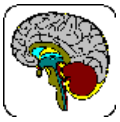
1.3.b Mental imagery



Mental imagery is supposed to be a basic phenomenon of all conscious mental operations, which is carried out in any sensory modality. It is considered to be an important component of conscious experience and of substantial functions in the mental process like associative learning and memory. Kosslyn described [Kosslyn, 1988] mental imagery as consisting of brain states like those during perception, yet arising without an appropriate sensory input; based on neurophysiological studies, Freeman [Freeman, 1983] came to similar conclusions. Most people would agree with the statement that mental imagery usually has a lower intensity compared to real perception and that it can be evoked and manipulated voluntarily. Furthermore, it is likely that certain psychopathological phenomena (e.g. hallucinations in schizophrenic patients) are based on the same neurophysiological mechanisms as mental imagery [Hoffman, 1986]. The investigation of mental imagery appears, therefore, to be of paramount interest for understanding physiological and pathological cognitive brain functions [Fallinger et al., 1997].

One ingenious and well-known experiment is the mental rotation task, which clarified some fundamental aspects of mental imagery [Shepard and Metzler, 1971] including localized changes in brain electrical activity. Paivio's dual-coding theory [Paivio, 1969] has also found support from studies which correlated its two cognitive modalities with biological measures. It has been shown that brain electrical activation patterns during visual imagery differ from those during abstract thoughts. Petsche et al. investigated the coherence of five EEG frequency bands during the visualization of an abstract concept as well as during the interpretation of a painting [Petche et al., 1992]. They found that thinking with language was related to left frontal activity, whereas thinking with images involved the activation of right frontal regions. Visual imagery has been mostly attributed to the occipital lobe including primary and secondary visual cortical areas. From a psychophysiological approach, Segal and Fusella showed that visual perception interfered with visual but not with auditory perception [Segal and Fusella, 1970]. This was interpreted as evidence for the competing utilization of cortical resources during imagery and perception.

1.3.c Auditory imagery and inner speech



A recent MEG study [Numminen et al., 1998] showed that short, self-vocalized vowels transiently activated the speaker's auditory cortex around 100 ms after speech onset. Remarkably, these responses were delayed by 11 ms in the speech-dominant left hemisphere than they did in the right hemisphere. In contrast, when the subjects were merely listening to a tape replay of their own vowels, the latencies of the responses were symmetric. Thus, speaking aloud primes the human auditory cortex at a millisecond time scale by delaying reactions to self-pronounced, i.e. expected, speech sounds mainly in the speech-dominant hemisphere. Such motor-to-sensory priming of early auditory cortex responses during voicing may constitute one element of speech monitoring that is essential during speech acquisition [Numminen and Curio, 1999].

Through our lifetime we are exposed to "What You Hear is What You Said" (WYHWYS) events, meaning that the voice we most often hear is our own. A proper interaction between speaking and hearing is thus essential for both acquisition and performance of the spoken language. This repeated input can imprint our auditory system so as to optimally process our own utterances. In this respect, single unit recordings in the primary auditory cortex have identified specific response patterns related to replayed phonemes

[Steinschneider et al., 1994]. Distributed audio-vocal interactions have been implicated in aphasia, stuttering and schizophrenic voice hallucinations; however, paradigms for a non-invasive assessment of self-monitored speech and its possible dysfunctions are still rare [Curio et al., 2000].

2. On the Influence of Real-time Bio-feedback

Feedback about performance is increasingly said to play an essential role in skill development. Without feedback, the subject could not determine which internal states moved the cursor or produced an animation effect. Feedback provides the information that allows the subject to acquire control over the EEG signal and thus over the animation. At the same time, however, feedback could have other effects, good or bad, on performance. The multiple possible effects of feedback have been addressed in a variety of studies [Black, 1973], [Salmoni et al., 1984]. Some effects are lasting, which is to say that they affect learning. Others are transient in that they only affect performance. It is not clear whether feedback is needed or desirable in all phases of performance. Salmoni et al. suggested that excessive guidance can sometimes actually decrease learning. However, it seems evident that feedback should be as rapid as possible since delayed feedback degrades performance. Smith and Smith [Smith and Smith, 1987] suggest that an ideal human-machine system should provide instantaneous feedback. This theoretical ideal is not possible for EEG control since data processing procedures require certain finite number of EEG data samples (e.g. 100-200 ms) and additional time for computation and screen display (e.g. 10-30 ms).

McFarland et al. state that the short-term role of feedback did not significantly affect the overall performance of the subject population [McFarland et al., 1998]. Nevertheless, some subjects were affected by the removal of feedback; this was shown by the fact that performance no longer significantly correlated with performance during previous sessions in which feedback was present. This suggests that, depending on the individual subject, feedback can have positive or negative short-term effects on performance. A more accurate analysis of the results provided in the above-mentioned work indicates, however, that subjects who performed well if not best in the feedback training sessions became confused in later sessions with reduced or no feedback. In contrast, for those who performed poorly, even to extent that almost no control could be gained in training sessions, the effect of “free flight” (i.e. non-guided by any instructing feedback animation) was shown to slightly improve their performance.

Our experiments brought forth several longer periods where users seemed to gain significant or to some extent “complete” control over the cursor or animation. This was, in turn, promoted by the feedback animation correlated with the user’s intentions. Several BBCI-users reported that it seems to take only a moderate effort to reach this “good” performance level. This level then remains quite stable, since it is supported by the correct feedback, which is successive. Most users were able to find this “point of entry” simply by relaxing and not paying any attention to the current feedback state or to the game score. This disinterest put users into a state resembling the one from the training session, consequently allowing the learning machine to recognize the single-trials with better performance. In cases where the feedback animation was, for whatever reason, not correlated with user’s intentions, the user was instinctively trying to push herself/himself to perform differently (i.e. by straining their own intentions), which is the exact opposite position of the recognizing machinery. Consequently, this leads to an even worse performance of the classifier. The obvious suggestion on this matter is that the user needs to achieve the same relaxed state of being during the feedback

session as she/he had during the training sessions, and, to pay as little attention as possible to the results of the feedback animation.

Alternatively, or in addition, the feedback might produce EEG responses that interfere with the EEG control required for accurate control. For example, cursor movement in a wrong direction can elicit frustration, a response likely to be associated with generalized EEG desynchronization. Conversely, cursor movement in the proper direction might lead the subject to anticipate hitting the target, e.g. in the “Brain-Pong” feedback scenario.

Feedback animation should be composed of a few and simple objects that preferably avoid brisk movements, high-contrast blinking and unexpected conditions. A BCI user who pays too much attention to the details of the animation might become distracted. The wasted attention will then prevent the concentration on internal states such as relaxation or control command emission. Furthermore, feedback animations of BCI systems that analyze oscillatory EEG feature components as visual stimuli might affect the visual α-rhythm, which is similar in frequency to the motor µ-rhythm, and which in turn is detected at the central scalp locations. Such responses would introduce noise into the EEG signal used to control the animation. In some cases it would be helpful to reduce the sizes of the controlled objects or to display their movement only partially. A smooth animation rather than brisk and unmotivated fidgeting has proven to be more convenient in the experiments conducted in the scope of the BBCI project.

In summary, feedback animation is essential and clearly necessary for learning EEG control. However, a feedback application designer should carefully choose the type of feedback the user will rely on. Any other redundant animation will introduce more noise to the EEG data, such that the resulting benefit deteriorates. Finally, the finding that subjects perform well (in some cases even better) without seeing any feedback indicates that, for the short term at least, extensive visual stimulation of the animation is not essential to maintain performance [McFarland et al., 1998].

Chapter III —

THE BERLIN BRAIN-COMPUTER INTERFACE (BBCI)

Die Überzeugung, dass man dem, was man nicht weiß, nachforschen müsse, macht uns besser und tapferer und weniger träge, als die Meinung, es sei nicht möglich, das zu finden, was wir nicht wissen, und man sollte deshalb auch nicht erst suchen
Plato (427 – 347 B.C.)

This chapter is the main part of the thesis and describes the overall design of the Berlin Brain-Computer Interface (BBCI). A description of the equipment used to assemble the BBCI will introduce the chapter, followed by important issues regarding data acquisition, assumptions on communication protocols, their main premises and features and the proper nomenclature for electrode positioning and naming. The experimental setup will be presented in detail before discussing the preprocessing procedures, classification methods and supplementary modules employed to extract as much relevant information as possible from the user's brain.

The concluding section discusses several feedback applications, which, at first glance, could be subdivided based on the control of one, two or more classes decoded from the data; or, which could also be divided into overlapping groups based on the control strategies the BCI system assumes for decoding the commands from the user's brain. I will try to cover all feedback applications that have been developed within the framework of the BBCI project complying with the classification practice of the BCI community, which divides feedback applications into the following three parts:

- User-instructive applications should be used in the first feedback-based BCI sessions. These applications provide the user with a feeling about the quality of signals extracted from the data and that.

- Brain-gaming applications can be employed in further feedback-based BCI sessions by users who have reached a significant level of intuitiveness in controlling a BCI; in turn, they can be employed in a BCI system once it has satisfactorily "learned" to assign classification results of the user data to command classes. These applications provide only very little, if any, instructive signal information; however, which abound with control reasoning and signification that allows the user to concentrate more on the aim of the game, namely achieving more credits. This kind of feedback applications can also be used in pediatric therapies.

- Convalescence and assistance applications can be used by advanced BCI users or patients who gained significant control over the BCI system. The user no longer has the possibility to assess the quality of the signal classification or the strength of the signal. She/he is expected to identify the feedback application by itself or certain objects affected by the control strategy as a part of the own imagination or as an absent body part. In spite of the finished development process for two of these applications presented in this thesis, exhaustive experiments are still to be conducted. Future research will investigate the aptitude of these applications to attain at least two objectives: (*i)* to restore control over a missing body part (e.g. after the amputation of a limb), albeit the control is only virtual, i.e. it actually remains separated by a robotic interface; (*ii)* to alleviate or eliminate phantom pain that resides in the sensory brain centers responsible for the absent body part.

Figure 7 depicts the abstract overview structure of any BCI system and the BBCI system in particular; a structure on which the setup of this chapter is based. Please note the coloring of the different modules; this coloring scheme will be reused in all successive figures of this thesis.

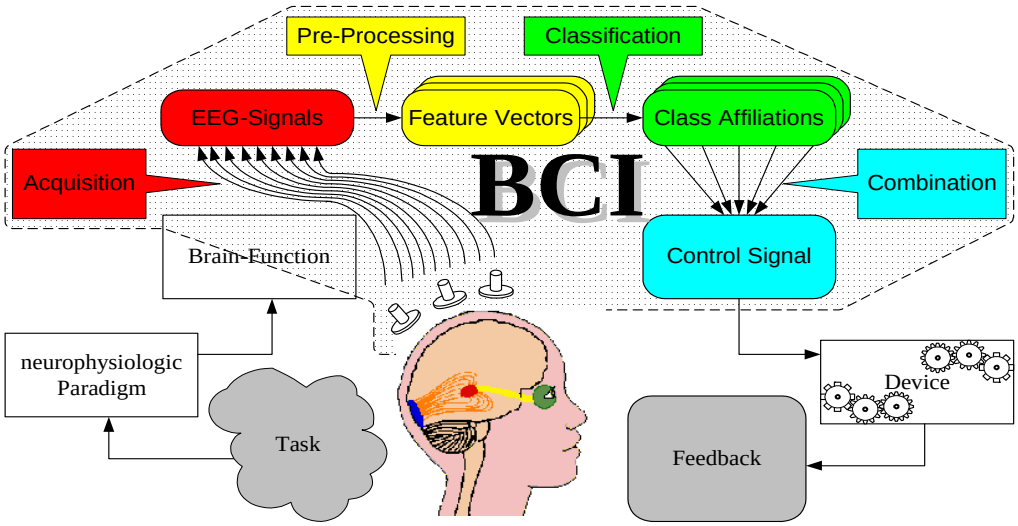


Figure 7: The abstract overview of any BCI system and the BBCI system in particular. A BCI system relys on a neurophysiological paradigm, processes acquired data through several stages and constructs a command to control a device or application. The user finally observs the results of this control perceiving the feedback information.

As it is common in most computer applications, the center of interest is the user who is given a task to perform. The neurophysiological paradigm of the given task will then induce a brain function; thus, it is not the user, directly, who induces or influences the brain function. The EEG signals are then acquired with the help of hardware elements such as electrodes, wires or amplifiers. The pre-processing procedure transforms samples selected from this continuous data and generates high-dimensional feature vectors that highlight the most important characteristics of the corresponding neurophysiological paradigm. Feature vectors are then classified by a machine-learning approach. Several similar or different pre-processors and classifiers may be employed that can make the system more powerful. For this purpose, a combiner is employed to merge the results of all classifiers and to produce a control signal that is then transmitted to an arbitrary device. This could be a wheelchair, a neuroprosthesis or, more abstractly, a computer simulation of such a device that provides the user with the feedback or bio-feedback information. This, finally, closes the loop to the user. The system we refer to as a Brain-Computer Interface is enclosed in the dotted area. However, the design of appropriate tasks, the investigation of new neurophysiological paradigms and corresponding brain functions, as well as the development of convenient feedback devices and their simulations and beneficial control strategies – all these satellite issues are paramount for BCI research and development and the future integration of such systems into the potential user environment.

I will start this chapter's sections from the hardware angle and follow the main flow of an information unit through to the BBCI system – correspondingly shown in the above figure – until the information loop is closed with the feedback application acting on user perception devices. Please note that a general-purpose BCI system – and the BBCI is one – does not depend on a single specific, either neurophysiological or any other, paradigm and should therefore be regarded as an abstract human-computer interface.

1. Hardware

Less than a decade ago, when patients needed an electroencephalogram (EEG), they were hooked up to a big cabinet-like machine normally set up in a separate chamber next to the consulting room. This machine amplified some few channels of weak signals acquired from the surface of the patient's head and then recorded these onto a sheet of endless paper. A subsequent series of EEG recorders were equipped with large rolling tables (see Figure 8); yet these also recorded the curves on paper. Printed paper graphs had the drawback that they could only be evaluated through a mere visual reading on the part of the physician; thus, their entire data could not be fully exploited.

The above-mentioned techniques were renamed as "analog EEG" once the more powerful and practical "digital EEG" techniques evolved; these quickly gained predominance not only in the clinical setting but also in research. This fortunate transformation of EEG acquisition hardware was, of course, due to the rapid development in computer architecture and software technology.

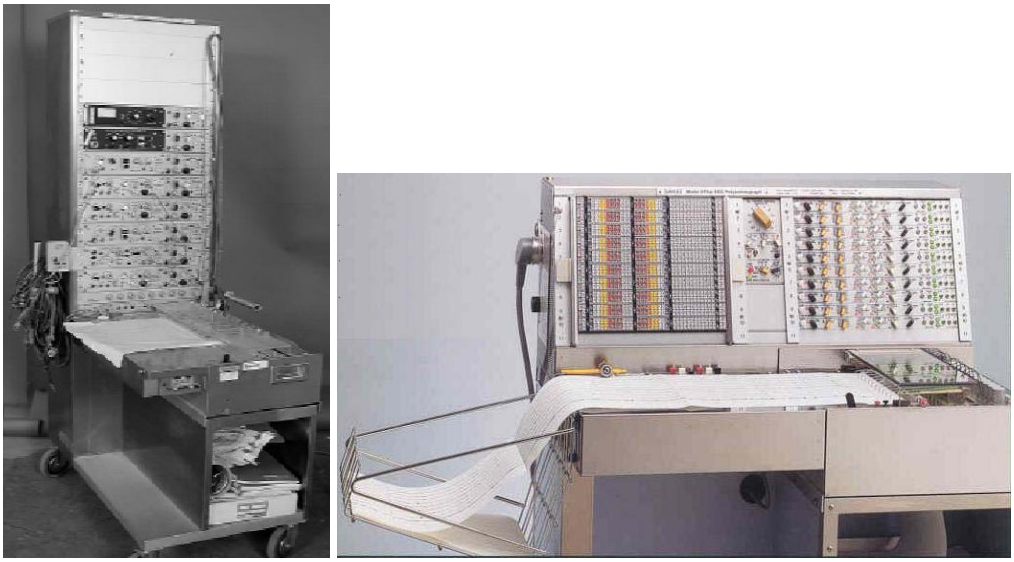


Figure 8: Historical electroencephalographs: Grass 1 (left) amplified 6 channels of EEG. Grass 2 (right) able to process up to 10 channels and to perform a simultaneous frequency analysis. From the Web site of the manufacturer

1.1 Electrodes

As outlined above, the BBCI technology for data acquisition – electroencephalography – employs surface electrodes placed over determined regions of the user's head. These electrodes are attached onto a Brain Cap at certain positions (see Section 2.1f.) and consist of an upper and lower electrode cap made of isolating plastic. The lower electrode cap contains the heart of each electrode – the electrode plate (Ø is 7.5 mm), an alloy of `Ag` and `AgCl`. A centered circular through aperture of about 2.5 mm in diameter allows the experiment conductor to fill in electrolyte gel using a syringe. This serves to establish a good electrolytic conductivity between the epidermis of the user's head and the electrode plate. Finally, a low-impedance connecting wire is soldered on the upper side of the electrode plate; this wire can carry extremely low currents to the head-boxes. Its isolation and shielding reduces its sensitivity to interfering noise currents from the surrounding environment. The assembly of a typical EEG electrode is illustrated schematically in Figure 9.

Other "contactless", dry electrodes, based on capacitive measurement techniques, are currently the state-of-the-art in electrical engineering [Harland et al., 2002] and promise valuable results for future BCI systems. This means that electrolyte gel will no longer be necessary, meaning also that users will no longer need to wash their head after each experiment. In addition, it will no longer be required to fill each of the 128 electrodes with electrolyte gel before each experiment; this will save considerable time.

The 128 connecting wires are bound into four flat cables and connected to two head-boxes. Certain electrodes can then be assigned to corresponding data transmission channels, which are then consecutively numbered. Four flat cables connect each of the head-boxes with a corresponding amplifier.

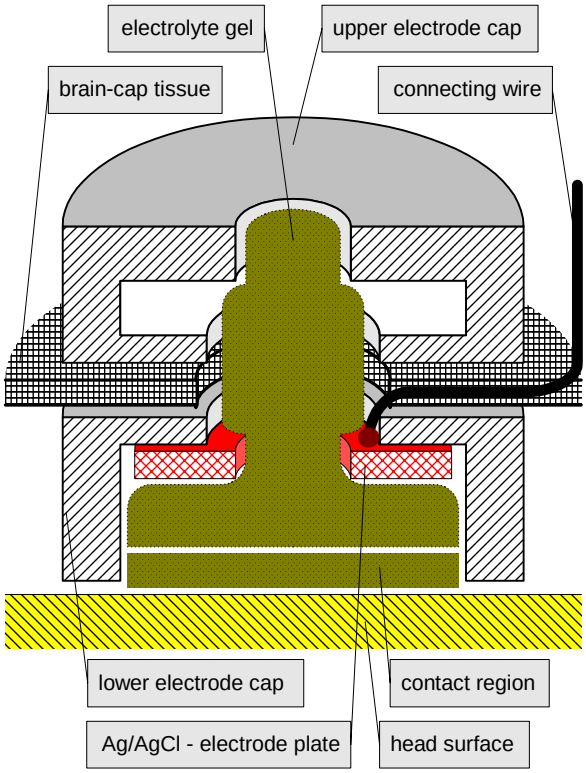


Figure 9: Schematic assembly of an electrode. The Brain Cap's fabric is enclosed between the upper and the lower plastic caps. In the latter an electrode plate with a connecting wire is mounted. Electrolyte gel is filled into the electrode through the aperture to assure low impedance between the epidermis and the electrode plate.

1.2 Amplifiers

The BBCI project employs four BrainVision© BrainAmp™ or BrainAmpDC™ EEG amplifiers obtained from the Brain Products® GmbH, Munich, Germany [`www.brainproducts.com`]. These amplifiers are employed in a variety of neurophysiological signal recording methods, including EEG, EMG, EOG, ERP, and SEP. A relatively high sampling rate of 5000 samples per second allows for the reliable acquisition of signals with a limiting frequency of 1 kHz. The amplification is done with a signal-to-noise-ratio (SNR) of 90 dB, a resolution of 100 nV, and within a range of ±3.2 mV. The amplified result is then digitized with an accuracy of 16 bits per sample by a built-in A/D-converter and transferred to the recorder PC. To minimize environmental and interference noise, an opto-coupled twin fiber-optic cable is used for the data transfer to a PCI interface card built-in to the recorder PC. Finally, each amplifier capable of processing 32 channels is contained in a 15 × 15 × 6 cm box (w × d × h), which weighs 1.2 kg. Figure 10 presents an amplifier as incorporated in our experiments.

Figure 10: BrainVision© BrainAmpDC™ amplifier employed for the amplification and digitalization of the EEG voltage of 32 channels simultaneously. From the Web site of the manufacturer

1.3 Recorder PC and the acquisition software

The recorder PC is a common Intel® Pentium™ 4 machine with 1.3 GHz and 1 GB of RAM and runs Microsoft® Windows™ 2000 Professional Edition. It contains two built-in PCI interface cards, each capable of acquiring signals from two different amplifiers. The installed software for data acquisition (BrainVision© Recorder™) was provided by the manufacturer of the amplifiers.

The acquisition software is capable of recording and visualizing up to 256 channels of continuous EEG, EMG or ECG data together with event markers (e.g. stimulus or response). Online averaging and segmentation in combination with online artifact rejection and baseline correction can be performed automatically during data acquisition. Moreover, several useful parameters (e.g. sample rate, gain, various filter characteristics, scaling and polarity) can be configured individually for each channel. A built-in online access to the currently acquired data via a well-defined TCP/IP-based communication interface allows for implementation of biofeedback paradigms like those addressed in this study. The communication protocol and the data format will be described in more detail later in Section 2.3.

2. Data Acquisition

This section gives an overview of the BBCI data acquisition approach. We will begin with the Brain Cap's and with other physical electrodes and their conversion into logical channels according to their positions and application domain. We will then discuss the communication protocol used in real-time online data acquisition and briefly illustrate its benefits for minimizing data loss and for decoupling the data processing system to comply with the assumed asynchronization of the system.

2.1 The Brain Cap

A Brain Cap is somewhat like a bathing cap, yet made of elastic fabric, and is furnished with 128 electrodes (see Section 1.1.). It is important that the Brain Cap encloses the user's head tight enough to prevent the desistance of single electrodes, while avoiding squeezing the user's head to the extent that she/he experiences discomfort. Several sizes of Brain Caps exist.

Additional electromyographic (EMG) electrodes are attached to the users' forearms (and, depending on the type of experiment, to the legs) over the corresponding actor muscle that is used to perform finger (or leg) movements in order to strike a key (or a pedal). An EMG is recorded in a bipolar way by attaching two electrodes along the actor muscle. The actual potential is then calculated by subtracting their individual potentials, canceling out the potential of the "Reference" electrode (*Ref*). This data is then used in later analyses as a reference to ensure the evidence of our classification approach. By performing similar calculations based on EMG activity, it can be shown that results are not due to muscle mobilization, neither in well-behaving nor in healthy users.

Further electrooculographic (EOG) electrodes are attached around the user's eyes and acquire the EMG activity of the muscles that control the position of the eyeball (needed for focusing a target on a computer screen or for controlling the user's gaze direction). Since EOG is a special type of EMG, it is recorded in a similar, bipolar manner. Two electrodes are attached on the temporal lobes so as to record horizontal eye movements, e.g. when the user is following a target moving aside. Further two electrodes are positioned above and below one of the eyes, so as to record vertical EOG and eye blinks. This data may be used in later analyses to clean up the EEG from interfering artifacts as well as to assure that the classification results from experiments with well-behaving and healthy users are not due to eye movements. EOG data is used also to recognize and prevent possible involuntary cheating, e.g. producing eye movements correlated with the task.

2.2 Electrode positions and nomenclature

Electrode positions are designated in the Brain Cap according to the extended international 10-20 system [Sharbrough et al., 1991]. It should be mentioned that the standard 10-20 system was developed over a decade ago, when recordings of no more that a dozen of electrodes have been sufficient; the standard system handles only 32 electrode positions. Later, it was extended to handle up to 64 electrode positions by introducing additional electrodes between the previously existing in the standard system. Modern BCI systems use up to 128 electrodes and are no longer a rarity. From its beginning, the BBCI project made a point of using as many electrodes as were available in order to achieve the highest possible spatial resolution. Additional electrodes were introduced to the extended 10-20 system by placing them between those of the extended 10-20 system; they were also named according to the 10-20 nomenclature by combining the labels of the neighboring once.

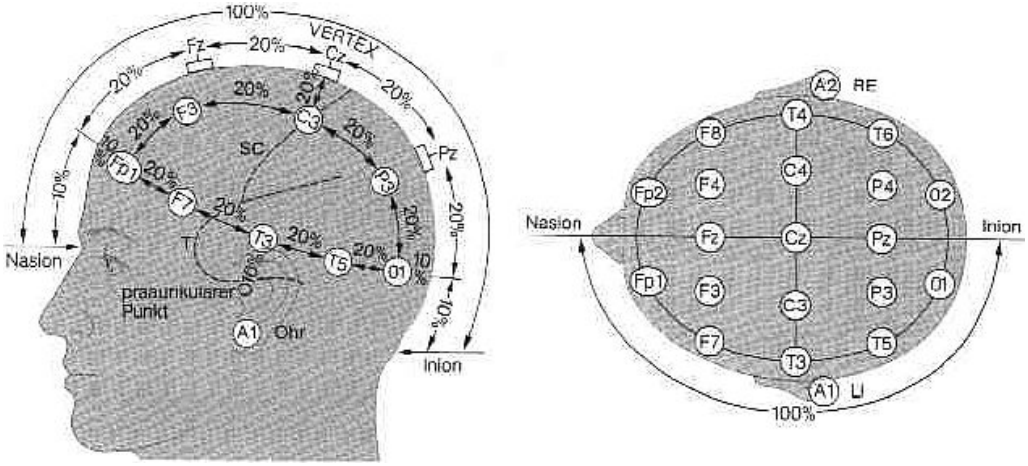


Figure 11: Standard international 10-20 system of electrode positions and the naming nomenclature. The system handles up to 32 electrode positions; a subset of 21 most important are shown. From [Sharbrough et al., 1991]

In order to optimize the fitting of the electrode cap, the user's head is measured. For this, the distance between the Nasion and the Inion is taken as one complete unit (100%). Half the distance between these two points then designates where the central electrode of the Brain Cap (`Cz`) should be placed over the vertex (see Figure 11). The other electrode positions are subsequently mapped onto the cap by drawing the main sagittal (anterior-posterior) lines (`F` and `P`) at a 20% distance, of the complete unit, in front of and behind the vertex, respectively comprising the frontal and parietal electrodes. Finally, the main, bottom electrode line is plotted a further 20% of the unit distance in front of the `F`-line and behind the `P`-line, which is exactly 10% above the Nasion and the Inion. Electrodes on the main bottom line are redistributed according to the same 10-20 scheme; the unit distance is determined by the length of the curve between the Nasion and the Inion (see top view of the scalp on the right of Figure 11). Scalp topographies, maps or electrode placement in medicine commonly depict the user's head viewed from above, with the nose at the top.

The labels of the electrodes are composed of capital letters and a number. Letters refer to anatomical structures, like Anterior (`A`), Frontal (`F`), Parietal (`P`), Occipital (`O`), Temporal (`T`) lobes or the Central (`C`) sulcus, while numbers denote sagittal (anterior-posterior) lines. Odd numbers correspond to electrodes on the left hemisphere and even numbers to electrodes on the right hemisphere. The numbers rise in proportion to the distance of the electrode from the central sagittal line. Electrodes located on the central sagittal line are marked by a small '`z`' concluding their label, rather than being indexed by `0`. Labels with one or two capital letters correspond to the 64 electrodes of the extended international 10-20 system [Sharbrough et al., 1991]. Labels with three capital letters were composed from the neighboring electrode labels and denote additional channels in a 128-channel setup. All EEG activity is measured against the Reference electrode, `Ref`, which is mounted on the Nasion; it is also common to mount the Reference electrode on the earlap. In both cases, the reference electrode should be mounted on areas that are void of EEG activity yet that are sufficiently close to other electrodes carrying EEG data. Some experimental setups require the reference electrode to be mounted at the position of `Cz`; in these cases, the `Cz` channel becomes void. The ground electrode, denoted as `Gnd`, is mounted on the forehead on the position of the `AFz` electrode. However, varying placements of the ground electrode does not affect significant changes in data accuracy. The positions of the electrodes and the labels of corresponding EEG channels are illustrated in Figure 12.

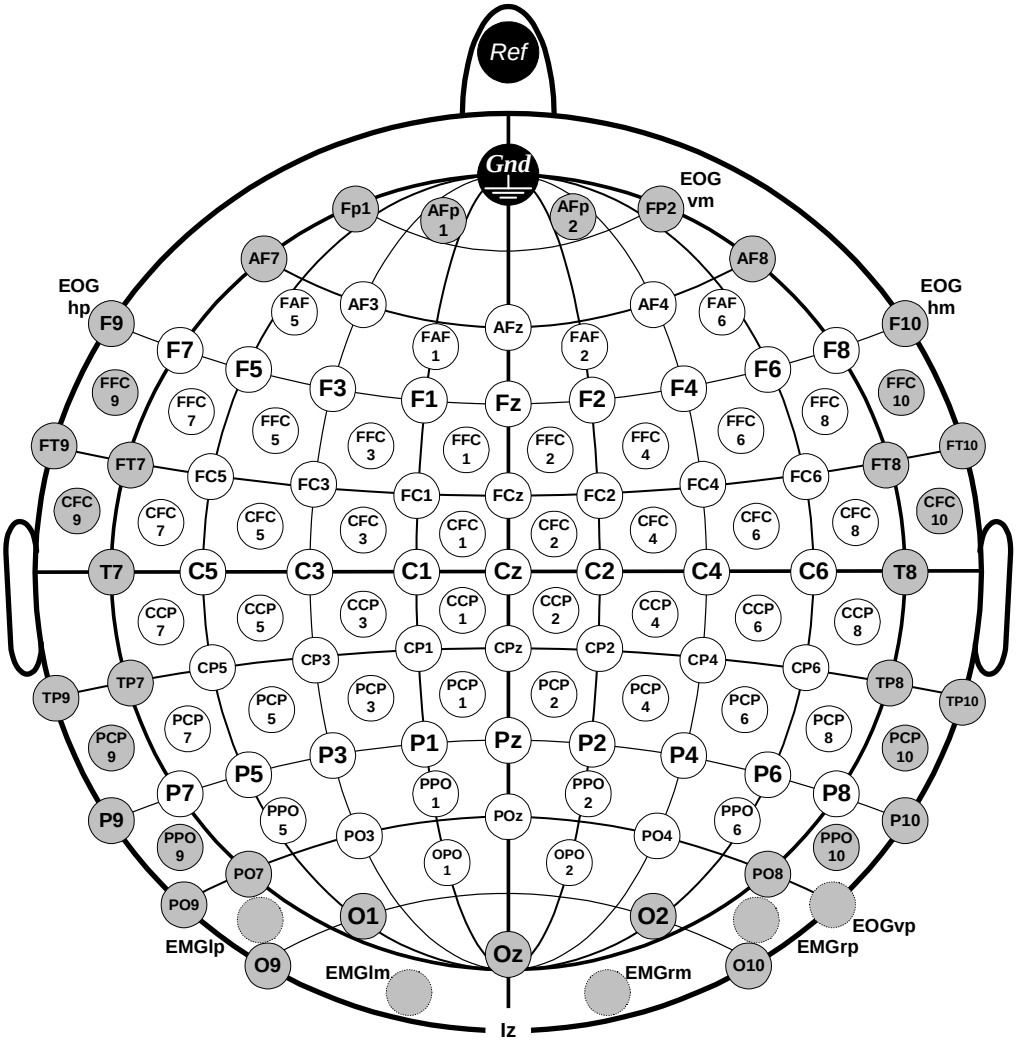


Figure 12: Positions of electrodes in BBCI experiments and naming nomenclature of the corresponding EEG-channels according to the international 10-20 system. Selected electrodes have been removed to be used for the acquisition of EOG or EMG. The Reference electrode is mounted on the Nasion; the Ground electrode is placed at the position `AFz`.

Some electrodes, like `F9`, `F10` and `Fp2`, because of their positions, are suitable for acquiring EOG activity such that the horizontal occulogram (`EOGh`) can be calculated directly during the acquisition from the difference between channels `F9` and `F10`. The vertical EOG employs an electrode positioned under the right eye and can be calculated by subtracting its value from the value acquired at the EEG channel `Fp2`.

Some EEG electrodes that are not taken to yield relevant information (e.g. at the occipital lobe or near the Inion) and therefore have been removed from the Brain Cap but are attached to the forearms (or the acting leg, depending on the type of experiment) over the actor muscle to record the EMG activity. This means that up to seven electrodes of the Brain Cap are not used for the acquisition of EEG but rather for various auxiliary control signals. At the BBCI project, most experiments were performed with about 120 EEG channels.

The voltage measured by the electrodes is very low and fluctuates rapidly within a range of about ±50 µV around a certain baseline. Electromagnetic noise from the surrounding environment (mainly 50 Hz or 60 Hz power outlet frequency) interferes with the data via connecting wires that act as small "antennas". To assure low impedances between the electrodes and the scalp (ideally below 5 $k\Omega$), each electrode is filled with electrode gel before experiments start.

2.3 Communication protocol

For the recorder software to achieve uninterruptible data acquisition, it needs to run in privileged mode and not be disturbed by other applications. Therefore, the implementation of this software is performed as efficient as possible such that the average processor load during the data acquisition process does not exceed 15%. Nevertheless, the system has its limits; if complex signal processing is carried out simultaneously to an additional privileged process on the same computer, all online activity may fail to function in real-time. This means that the data acquisition module will not be able to perform its task in time, resulting in late and irregular presentation of data blocks. This in turn affects the internal synchronization of the recorder software and could lead to a loss of data blocks and event markers. To avoid this risk, it is of fundamental importance to design the BBCI as a distributed system. A more detailed description of the software technology, distributedness and parallelism aspects is provided in Chapter IV.

The recorder software allows transmitting EEG data to other computers on Local or Wide Area Network (LAN/WAN) during the acquisition process. I refer to this feature in further explanations as Remote Data Access (RDA). It is based on a standard client/server architecture that transfers data according to the TCP/IP protocol. The recorder PC here acts as a server, allowing up to ten client PCs to be connected to it and registered for the reception of the currently acquired data blocks. In this way, a client program can be implemented in any programming language that supports socket-based Internet communication and that can be run under any of the mainstream operating system such as Windows, Linux, UNIX or MacOS.

3. Training Procedure

"Let the machines learn!"
Guiding motto of the BBCI (2001-present)

The leitmotif throughout the BBCI project is: "Let the machines learn!" Thus, the user should need only minimal training for operating it. And therefore, the training procedure, as described in this section, applies more to the training of the computer-based system than to teaching the subject, who has taken a cold-start in using the BBCI system. Nevertheless, the user is instructed by the experiment conductor about the task to be performed during the training session. These tasks could include real finger or body part movements, imagining these movements, imagining tactile sensation of a body part, performing a non-trivial arithmetical calculation or a 3D-figure rotation, spelling a non-trivial word, or listening to a melody. Depending on the experiment, the task may be performed on demand, i.e. queried by the system, or self-paced by the user. In system-queried experiments, the task module places an

event marker into the data or, more generally, transfers this information to the BBCI system. In self-paced experiments, the user performs an action that can be monitored and which therefore prompts the system to place an event marker in the acquired data.

Figure 13 illustrates a simple BCI training approach with two actors: a user generating brain signals (left), and the BBCI system (right) receiving these signals through the acquisition machine (top). The BBCI system is able to adjust its signal recognition parameters based on the event markers received either from the task module or stored together with the acquired data.

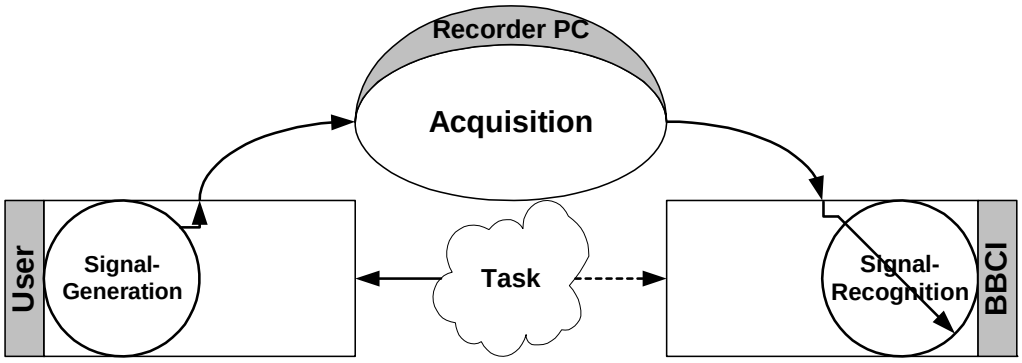


Figure 13: Machine training session of the BBCI system. The user (left) is performing the task either queried or self-paced and generates certain spatio-temporal pattern of EEG, which are acquired by the recorder PC (top) and transmitted to the BBCI system (tight) for training its signal recognition module.

3.1 Experimental setup

The experimental setup of all BBCI experiments is composed of two sessions: the machine training and the application (as it is exemplarily illustrated in Figure 14). The illustrated setup can be easily adopted for other experimental paradigms by applying only a few small changes.

In the first session, the machine training session, the user is instructed to sit comfortably and, as far as possible, to avoid any muscular artifacts (e.g. biting, swallowing, yawning, or moving the head, arms, legs or the entire body) as these would induce electromyographic (EMG) noise. This noise would interfere with EEG signals and significantly decrease the signal-to-noise ratio (SNR). The user receives a detailed explanation of the task and is given some time to adjust to the constraints. The system then acquires example EEG from the user while she/he performs a given task during the training procedure. The single-trial differential potential distributions of the Bereitschaftspotential (BP) (also known as Readiness Potentials) that preceded the voluntary movement over the corresponding contralateral primary motor cortex are analyzed using the multi-channel scalp EEG recordings. This results in an appropriate user model comprised of three parts:

- The pre-processing model. This model can be based on a parametric approach, e.g. a spatio-temporal matrix calculated from the data samples. This acts as an optimal filter for mapping the raw data to a feature space in which a classification can then be more easily performed. A calculation of the auto-regressive (AR) model parameters or rules used for calculating adaptive autoregressive (AAR) model parameters can serve as an example of

a parametric pre-processing model. However, we employed a non-parametric pre-processing model in most of our experiments (see Section 4). [Blankertz et al., 2003]

- The classification model. This model is usually based on a parametric approach of linear classification, yet can also include a kernel-based or a non-linear classifier. A matrix of weights is calculated from the preprocessed data. This matrix maps the data from its feature space to a set of one-dimensional spaces of fuzzy values that indicate the certainty of the classifier's decisions regarding the affiliations with the one or the other class of interest. The BBCI employs a linear classifier – the Regularized Fisher's Discriminant (RFD) (see Section 5) [Blankertz et al., 2002], [Krepki et al., 2004].

- The combination model. This model is limited to a set of logical rules that combine the fuzzy values of affiliations produced by the classifier model to a control command that is conform to the communication protocol of the feedback module. This command will then be used to control the animation of the feedback application. The BBCI employs a simple Winner-Takes-All (WTA) combiner module in most of its experiments [Waterhouse and Robinson, 1994].

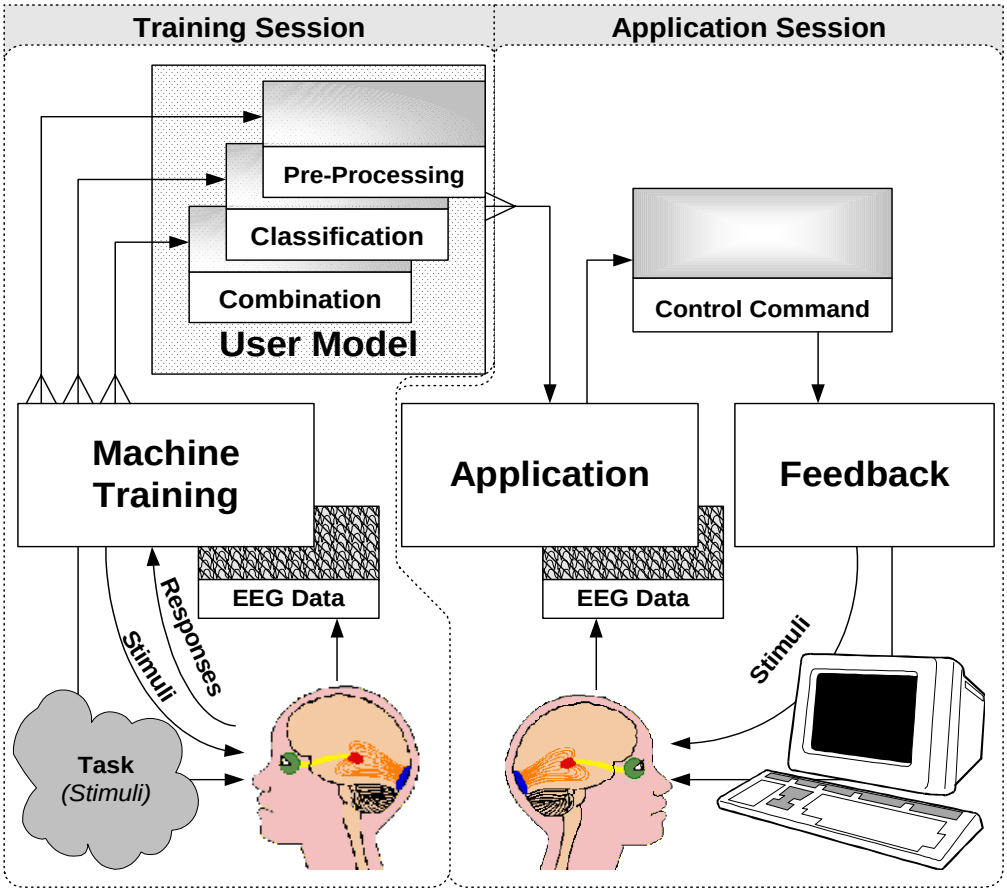


Figure 14: Exemplary experimental setup of a BBCI experiment. During the training session (left) the user performs a task and the learning machine setups the user model. During the application session (right) this model is applied to the generated EEG pattern aiming their recognition and transformation into command for controlling a feedback application.

In experiments that incorporate brain signals of healthy subjects executing real movements, a paradigm has to be able to predict the laterality of imminent hand movements prior to any EMG activity. This is necessary in order to rule out a possible confound with afferent feedback from muscles and joint receptors contingent upon an executed movement.

The BBCI is based on Lateralized Readiness Potentials (LRPs), which are a form of Slow Cortical Potentials (SCPs) and which appear during movement preparation. Interestingly, the intrinsic movement execution is not essential since LRP variants can also be observed for imagined movements in healthy test persons [Beisteiner et al., 1995]. However, the reader should note that a person's intention to move her/his amputated arm (phantom movement) is not identical with imagined arm movements of a healthy person. This is because in the latter case an additional "no-go" or "veto" signal is required to prevent the actual motor performance. BBCI is capable of recognizing both paradigms: imagined and executed movements, as described in further sections and validated by appropriate experiments. A more detailed neurophysiological description of paradigms can be found in Chapter II.

When the appropriate user model is set up, the experiment conductor can proceed with the second session: the application of the obtained model to the user's data (as illustrated in the right part of Figure 14). During this session, the communication loop is closed by providing the user with the visual, auditory or tactile feedback from the application. This feedback application can be run on a separate computer and is controlled with commands emitted from the BBCI signal recognition module. The user's task changes now from the "blind" execution or imagination of actions to the more complex but psychologically intuitive task of gaining control over the application. The system does not provide any additional stimuli for synchronizing the user's actions, nor does it require any intrinsic responses from the user – these functions have been shifted to the feedback application that is fine-tuned to the user's proficiency level.

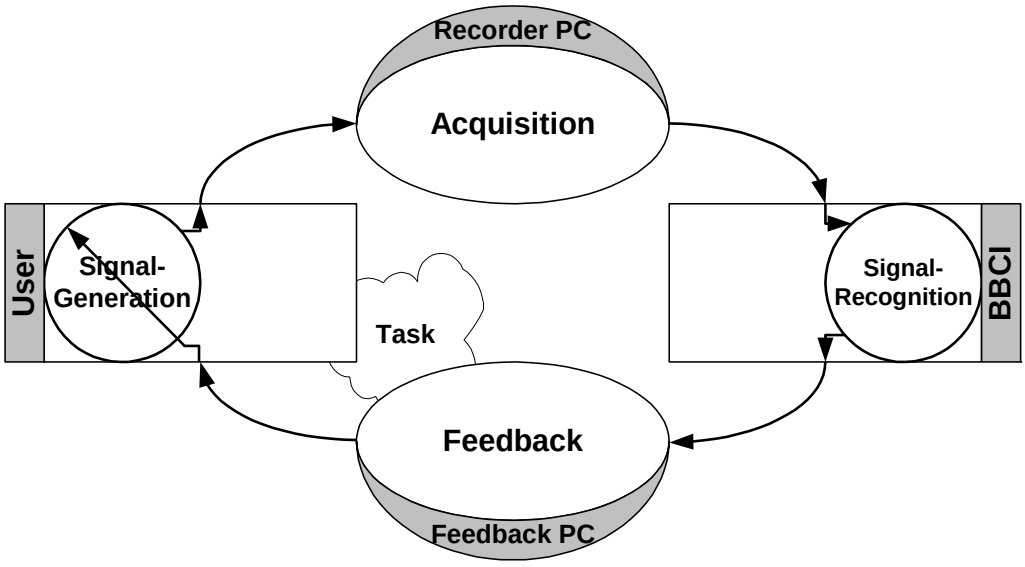


Figure 15: Application session of the BBCI system in a feedback-based experiment. The user performs a task (controlling the objects of the feedback applictaion) and generates certain EEG patterns that are acquired by the recorder PC and transmitted to the BBCI's signal resognition module. The BBCI sytem in turn emits control command to the feedback application.

As indicated in Figure 15, during the application session, the task is given the role of a logical interface between the user and the feedback application. The user's "signal generation module" – the brain, as a plastic entity – and the control strategy employed by the user can now be updated and enhanced by the information provided from the feedback application. Therefore, the BBCI's signal recognition module – the user model – remains unchanged throughout the complete session. Nevertheless, it is conceivable to integrate innovative techniques of active learning, allowing for minor updates of the user model [Castillo and Wrobel, 2002]. Such techniques would advance BCI systems as a whole by turning them into bilateral learning systems, which would then consist of two flexible entities adapting to each other.

- The user can update her/his signal generation policy and vary the control strategy to improve control over the feedback application. This in turn induces permanent changes in the brain signals presented to the acquisition machine.

- The BBCI can continuously update its signal recognition module – the user model – to better fit the control strategy currently employed by the user, which aims providing the user with a higher quality of feedback control.

For this purpose, an additional communication channel is needed to transfer a reinforcement signal to the BCI system. This will be discussed together with error signal detection in Section 6.3 and with the dedicated feedback application in Section 7.5.

The task to be performed by the user during the training session could be, among others, a self-paced real finger movement or a machine-queried imagined finger movement. They will be described in the following.

3.1.a Self-paced real finger movements

This kind of experiment can be performed with healthy users who have motor control of their extremities, e.g. fingers. The users are instructed to place their hands on the computer keyboard (German layout) in the initial 10-finger-typing position. Their left and right index fingers should then be placed over letters '`F`' and '`J`' respectively and their left and right pinky fingers correspondingly over letter '`A`' and 'Ö'. Their arms, hands and fingers should be as relaxed as possible and they should not move forwards, backwards or sideways.

Typing on a computer keyboard using the left or right index or pinky fingers must be performed with a certain frequency, e.g. about 0.5, 1 or 2 Hz, as it is specified by the experiment conductor yet self-paced regarding tact and sidedness. The experiment conductor can adjust the user's actions with short and precise commands, enforcing a faster or slower tact in order to retain the inter-tap interval as constant as possible. The conductor can also increase or decrease right or left repetitions or alternations in order to keep the transition matrix well-balanced. An unbalanced distribution matrix (e.g. containing a lot more left-trials than right trials or vice versa, or indicating a lot more repetitions of the same movement than alternations or vice versa) may invalidate the classifier training efforts. This is because in the training procedure it could become evident that a classification based on the post-trial data of the previous trial is useful for predicting the class of the future trial. The reader should note here that trial intervals are selected from the continuous data as causal windows, i.e. they only contain data points from the past. The experiment conductor must ensure a well-balanced transition matrix such that information from the previous trials does not contribute to the training procedure.

3.1.b Machine-queried imagined finger movements

Experiments with machine-queried imagined finger movements can be performed with neurologically diseased users as well as with healthy users (as described in Section 3.1.a).

However, the experiment requires some user training to prepare the movement and to emit the movement command without executing it intrinsically. A subsequently emitted “no-go” or “veto” command is needed for this purpose. Machine-queried imagined finger movements are also commonly employed in experiments with people suffering from severe neurological disorders, for example, persons paralyzed by spastic paraplegia, brain stem injury or amyotrophic lateral sclerosis (ALS), or those who lost extremities due to amputations. In these cases, users perform the “phantom movement” with their damaged or absent arm, hand or fingers. The brain signals of healthy users differ from those performing phantom movements. However, because the corresponding motor area of the cortex is still intact and because the underlying paradigm is identical for all humans, these signals are detectable by our signal processing approach.

Imagined or phantom typing has to be queried externally because we cannot gather information about the oncoming of the action or about its laterality. Two types of queries are introduced to the user:

- Early visual determination of the laterality of the upcoming action. This is indicated on the computer screen facing the user during the training session with big bold letters ‘`L`’ or ‘`R`’ or with arrows pointing to the left or right in the center of the screen.

- Auditory beat with a constant frequency, e.g. 0.5 Hz. This is produced by a digital metronome and indicates when the action determined by the foregoing visual query is to be initiated.

A more sophisticated query technique based on auditory stimuli only is conceivable and could be appropriate for unilaterally paralyzed users. It would be constructed on the basis of two different types of metronome beats, e.g. “tick” and “tack”, alternating sequentially. When hearing “tick”, the user is instructed to pay attention to it, but not to initiate any movements. When hearing “tack”, she/he is to either (*i)* execute an action with the intact extremity on the computer keyboard, which will place an event marker into the data; or (*ii)* generate a phantom movement, which will obviously leak the expected event marker.

All visual and auditory stimuli presented to the user during the training session will place time-sharp event markers in the data. Through this, training samples can be correctly extracted from the data and correctly affiliated to a certain class of interest.

3.2 Extracting training samples

The training is performed in 3 to 5 sessions of about 7 minutes each, interrupted by short breaks of user-defined duration (see Figure 16). Each training session is introduced by a 40-second and concluded by a 20-second relaxation period and includes a total of 6 minutes during which the user performs the given task repeatedly. A message on the monitor informs the user of the current period, i.e. relaxation or action. During the action periods, the user is provided with a static, thin fixation cross in the center of the screen; she/he is instructed to direct her/his gaze on it and to avoid ocular artifacts as much as possible.

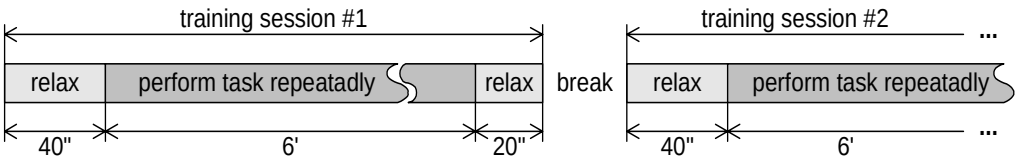


Figure 16: The setup of training sessions of the BBCI experiments. Each session lasts over 7 minutes including 6 minutes of repeatedly task execution. Break of user-defined duration are inserted between the sessions.

Depending on the frequency of the repeated action (defined by the experiment conductor), the data of one training session consists of 180-700 trials, namely those announced by the event markers. While in experiments investigating real finger movements the corresponding event marker, i.e. user response, carries all information about the actual onset of the action as well as its laterality coded in the marker label, the onset and the laterality of the action in experiments investigating imagined or phantom movements (see 3.1.b) is coded in two consecutive markers, i.e. user stimulus. This makes it necessary to combine these to a single one according to logical rules before starting sample extraction. Let us define an event marker as:

$$\text{Marker} := \left(\mathit{Type} : \text{enum}\left\{ \mathit{Stimulus} \mid \mathit{Response} \right\},\ \mathit{Label} : \text{String},\ \mathit{Position} : \text{UINT} \right), \tag{2}$$

where *Type* can be *Stimulus* or *Response*, the *Label* carries the description of the marker and *Position* defines its temporal position in the continuous data. A combination rule may then be assembled as:

$$\forall m_1, m_2, m_{new} \in \text{Marker} : \wedge \begin{bmatrix} m_1.\mathit{Type} = m_2.\mathit{Type} = \mathit{Stimulus} \\ m_2.\mathit{Position} - m_1.\mathit{Position} < \tau \end{bmatrix} \Rightarrow \\ \Rightarrow m_{new}\left(m_1.\mathit{Type},\ m_1.\mathit{Label},\ m_2.\mathit{Position} \right) \tag{3}$$

which will replace markers m_1 and m_2 with a combined marker m_{new}, where τ is a time constant indicating the maximum allowed duration between the two consecutive markers (e.g. which may indicate the determination and detection queries).
Each class of interest covers its own sample selection parameter set defined as:

$$\mathit{SSP} := \left(\left\{ \left(\text{Marker}.\mathit{Type},\ \text{Marker}.\mathit{Label} \right) \right\},\ n,\ t_d,\ t_i \right) \tag{4}$$

where a set of *Type*-*Label* combinations identifies the affiliation of markers to a certain class of interest, *n* gives the number of training samples to be selected from the data, and t_d and t_i are time constants indicating the delay of the initial sample and the inter-sample interval. These samples can be figured as temporally overlapping spatio-temporal sub-matrices of the continuous data matrix and indicate time regions where information about task performance is expected to be coded. Please find these so-called *Action* samples, denoted as `1a`, `2a` and `3a`, in Figure 17, where `w` denotes the width of the extracted training sample and is fixed for each preprocessing procedure.

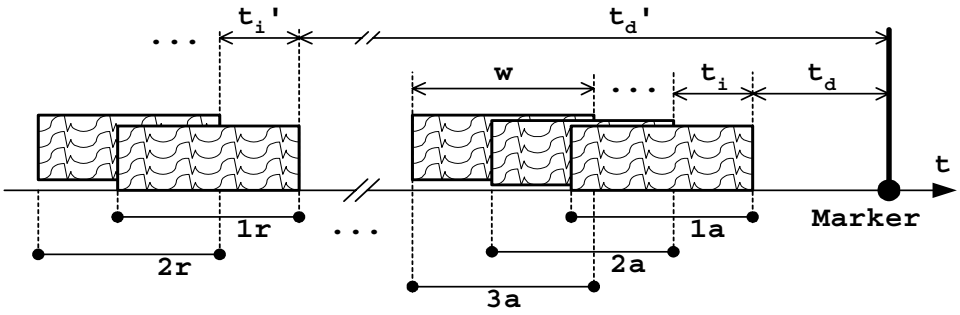


Figure 17: Sample selection procedure for 3 *Action* and 2 *Rest* training samples. A number of samples are selected relative to the event marker specified by the parameters of the SSP. *Action* samples (`1a`, `2a`, `3a`) are expected to contain EEG of movement preparation, which in turn should be absent in *Rest* samples (`1r`, `2r`).

In order to make the classifier invariant to the temporal shifts, it is useful to extract several training samples that belong to a single event marker from the continuous EEG data. The user may press the button or imagine the movement execution slightly shifted relative to the stimulus marker such that it cannot be determined exactly when the most prominent EEG sample is located. Several training samples instruct the classifier training procedure to enlarge the duration of the action window preceding the movement intension. However, extracting too many training samples has the drawback of including those without any neural activity into the "action" data set. Consequently, the training procedure will fail to find a satisfying classifier model. While the calculation of a satisfying classifier model is most often possible, it is a common observation for real-world data, that a perfect classifier is improbable.

Aside from the classes introducing *Action* samples, an additional class introducing *Rest* samples can be defined. This is useful if the BCI communication is to be asynchronous, i.e. if the user may fire a command at any time during the feedback application session and not only when she/he is queried by the feedback machine. In this case, the BCI system must be able to distinguish two kinds of processes: (*i)* the detection of an upcoming action and (*ii)* the determination of which action is coming up. For this purpose two parallel processing paths are initiated, both of which can employ different user models as well as different meta-models set up individually based on different sets of samples. The meta-model is given by selecting the appropriate pre-processing and classification procedures. Please note that the BBCI cannot perform the meta-model selection automatically. This means that the meta-models for both paths need to be determined by the experiment conductor; yet, the selection is a simple operation.

To identify the user model of the "determination" path, only *Action* samples of each class of interest selected from the data are used. The features extracted from these samples are set into opposition to each other. For example, in a two-class experiment (e.g. left vs. right finger movement), a sample set affiliated with the class of left movements is set into opposition to the sample set affiliated with the class of right movements, and vice versa. In multi-class experiments, the opposition structure depends on the meta-model involved such that several possible variants become evident and should ideally have been validated for feasibility in preceding offline studies. Several oppositions that proved to be useful for a 3-class experiment are summarized in Table 3 (completeness is not guaranteed).

Table 3: Opposition techniques that proved to be useful for a 3-class experiment

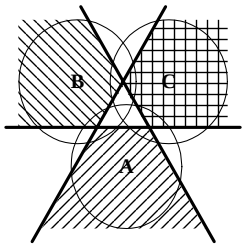
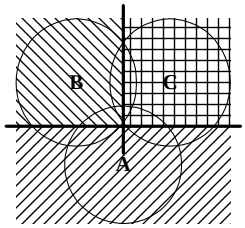
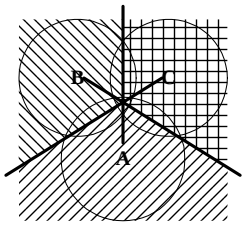
Var	Oppositions	Graphic representation	Comments
1	$A \leftrightarrow B \cup C$ $B \leftrightarrow C \cup A$ $C \leftrightarrow A \cup B$		This technique should only be used when making an error is too costly. Regions that may contain erroneously classified samples (unfilled) are labeled "unknown".
2	$A \leftrightarrow B \cup C$ $B \leftrightarrow C$		This technique is useful if a single class (here A) is distinguished by its special properties and the two other could be viewed as similar.
3	$A \leftrightarrow B$ $B \leftrightarrow C$ $C \leftrightarrow A$		This technique does not assume prior knowledge about any of the three classes.

Table 4: Appropriate (default) values for the sample selection parameter (SSP) set.

Aspired pace frequency	Action			Rest		
	`n`	t_d [ms]	t_i [ms]	`n'`	t_d' [ms]	t_i' [ms]
0.5 Hz	4, 5	-120	-40	3, 4	-800	-200
1.0 Hz	3, 4	-100	-30	2, 3	-400	-100
1.5 Hz	3	-90	-25	2, 3	-300	-80
2.0 Hz	2, 3	-80	-20	2, 3	-250	-60

To identify the user model of the "detection" path, samples of the *Rest* class are extracted from the data; this class's features are contained in its own SSP set. This procedure generates training samples (denoted as `1r` and `2r` in Figure 17) which are, irrespectively to their marker label affiliation, used in opposition to the union set of samples affiliated to all *Action* classes. Special attention must be paid in fast-paced experiments that samples of the *Rest* class do not intersect with samples of the *Action* class of the preceding event marker, as they should not

include any information about action. The conceivable and appropriate values for the SSP, summarized in Table 4, may be of interest to the reader interested in technical details.

4. Preprocessing Procedure

The main BBCI focus is on control applications, such as "virtual keyboard typing", that can be conceived as potentially resulting from the natural sequence of motor intention, followed by preparation and completed by the execution. Accordingly, the neurophysiological approach aims to capture EEG indices of the preparation of immediately upcoming motor actions. This thesis will present the BBCI's capacity to exploit the Bereitschaftspotential (BP), also called Lateralized Readiness Potential (LRP) in other publications. BP is a special case of Slow Cortical Potentials (SCP); they are slow negative EEG shifts that develop over the activated motor cortex prior to the actual movement onset for the duration of approximately one second. It is assumed to reflect mainly the growing neuronal activation (apical dendritic polarization) in a large ensemble of pyramidal cells. Previous studies [Lang et al., 1989], [Cui et al., 1999] have shown that in most subjects the spatial scalp distribution of the averaged BP correlates consistently with the moving hand, where the focus of brain activity is located contralaterally to the performing hand. For more detailed neurophysiological information, please refer to Chapter II.

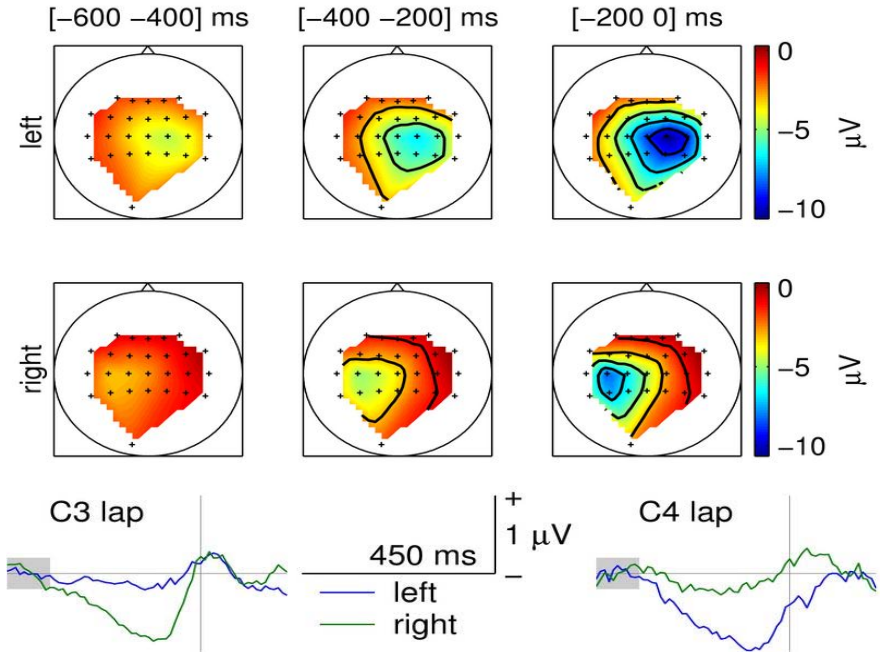


Figure 18: Scalp distributions of the Lateralized Readiness Potentials (LRPs) averaged over many single-trials of only left-hand finger movements (upper row) and only right-hand finger movements (middle row) within the three time periods relative to the movement execution. Laplace filtered EEG curves (lower row) averaged for left and right trials separately illustrate greated negativation over the motor cortex contraletral to the performing hand.

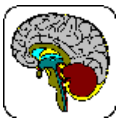


Figure 18 shows the Laplace[1] filtered EEGs around the left and the right motor cortices, i.e. around electrodes `C3` and `C4` within a time range of [-450 : 200] ms relative to the key tap, averaged selectively for left-hand-only and right-hand-only taps. The grey bar indicates a 100 ms baseline period during which an average value is calculated and subtracted from all data points in order to locate the graph around the abscise. The LRP lateralization is clearly specific for left, respectively right-hand finger movements, with right-hand finger movements producing a significantly higher negativation on the left hemisphere (near electrode C3, see left plot) than left-hand finger movements. Conversely, the negativation on the right hemisphere (near electrode C4, see right plot) is higher for left-hand finger movements than for right ones. The reader should note that these plots were produced by averaging over many single trials, thus summing out uncorrelated noise. However, this noise (e.g. acoustic, haptic, visual or olfactory stimuli or mental distraction) from the influence of the surrounding environment on the user has much higher amplitude compared to the data and is still present in each single trial; it therefore remains difficult to predict the laterality based on a single observation.

4.1 Feature selection procedure

BBCI is capable of any pre-processing procedure (i.e. for selecting the most prominent features) that matches the communication protocols: (*i)* it should be able to handle the continuous EEG data of a given format; (*ii)* it should result in the pre-processed data in a manner usable for further classification, i.e. in a special data format. Both interface data formats are presented in Chapter IV. In this subsection, I will describe the Feature Selection procedure as proposed by Blankertz et al. [Blankertz et al., 2002] [Blankertz et al., 2003] and integrated into the BBCI system [Krepki et al., 2003], [Krepki et al., 2004].

To extract relevant spatio-temporal features of slow brain potentials, we sub-sample signals from all or a subset of all available channels and take them as high-dimensional feature vectors. In pre-movement trials, extracted from the continuous EEG data as illustrated in Figure 17, most information is expected to appear at the end of the given interval; we therefore apply a special treatment to all training and test samples. Let us denote the raw data as an array *D* of *#D* elements acquired at sampling frequency *F*, and the array *d* containing elements sub-sampled to frequency *f*. Consequently, d consists of $w := \#D \cdot \frac{f}{F}$ elements (for both arrays the indexing starts at `0`). The samples are extracted – exemplarily, each of a length of *#D* = `1280` time points and sub-sampled to the frequency *f* = `100` Hz – from initially acquired data with a sampling frequency of *F* = `1` kHz, by applying the sub-sampling procedure (e.g. simply taking every 10^{th} data point). This provides us with epochs of *w* = `128` data points. This length is a fixed parameter for a fixed pre-processing procedure (compare Figure 17). Various sub-sampling procedures have been implemented and tested; see Table 5 for a summary. However, all yielded same or similar results; thus we limited this procedure to the one which incurred the least computational costs in terms of time/resource consumption. The following table depicts operations to be performed for calculating each data point of *d*, given *D* as well as the corre-

---

[1] Laplace spatial filter subtracts the mean value of all the neighboring channels from each of them separately. This action accentuates local activation in channel data. The calculation of the mean can be implemented in different ways, e.g. by weighting the neighbor's values with some spatial distribution function, and by certain useful definition of the measure of distances between channel positions. In our case this measure is given by the Euclidean distance between the electrode positions of the Brain Cap montage.

sponding computational costs; the operations are listed in the order of the number of arithmetic operations necessary.

Table 5: Sub-sampling procedures and the corresponding computational costs

Procedure	Calculation	Computational costs
First	$\overset{\#D\cdot\frac{f}{F}-1}{\underset{i=0}{\forall}} d[i] = D\left[i\cdot\frac{F}{f}\right]$	$O\left(\#D\cdot\frac{f}{F}\right)$
Last	$\overset{\#D\cdot\frac{f}{F}-1}{\underset{i=0}{\forall}} d[i] = D\left[(i+1)\cdot\frac{F}{f}-1\right]$	
Middle	$\overset{\#D\cdot\frac{f}{F}-1}{\underset{i=0}{\forall}} d[i] = D\left[\left\lfloor\frac{F+2Fi-f}{2f}+\frac{1}{2}\right\rfloor\right]$	
Random	$\overset{\#D\cdot\frac{f}{F}-1}{\underset{i=0}{\forall}} d[i] = D\left[Rand\Big\vert_{i\cdot\frac{F}{f}}^{(i+1)\cdot\frac{F}{f}-1}\right]$	
Mean	$\overset{\#D\cdot\frac{f}{F}-1}{\underset{i=0}{\forall}} d[i] = \frac{f}{F}\cdot\sum_{k=i\cdot\frac{F}{f}}^{(i+1)\cdot\frac{F}{f}-1} D[k]$	$O\left(\#D\cdot\left(\frac{f}{F}+1\right)\right)$
Next to Mean	$\overset{\#D\cdot\frac{f}{F}-1}{\underset{i=0}{\forall}} d[i] = D\left[\underset{i\cdot\frac{F}{f}\le j<(i+1)\cdot\frac{F}{f}}{\arg\min}\left(\left\vert D[j]-\bar{D}_i\right\vert\right)\right]$, with $\bar{D}_i = \frac{f}{F}\cdot\sum_{k=i\cdot\frac{F}{f}}^{(i+1)\cdot\frac{F}{f}-1} D[k]$	$O\left(\#D\cdot\left(\frac{f}{F}+2\right)\right)$

Assuming $f \le F$, and `#D` positive, yields:

$$O\left(\#D\cdot\frac{f}{F}\right) < O\left(\#D\cdot\frac{f}{F}\right)+O(\#D) < O\left(\#D\cdot\frac{f}{F}\right)+O(\#D)+O(\#D)$$
$$= \qquad\qquad = \qquad .$$
$$O\left(\#D\cdot\left(\frac{f}{F}+1\right)\right) \qquad O\left(\#D\cdot\left(\frac{f}{F}+2\right)\right) \tag{5}$$

Consequently, the sub-sampling procedure “Last” is expected to be the most efficient one with respect to both: accuracy of classification results and resource consumption. The same extracting and sub-sampling procedure is performed for all or a subset of available EEG channels separately.

To emphasize the late signal content, the data points of the sample window (which exemplarily ranges from -1400 ms to -120 ms (t_d = -120 ms) relative to the timestamp of the desired event marker) are multiplied by a one-sided cosine function `win[·]` defined in its discrete form in (6):

$$\forall_{i=0}^{w-1} win[i] := \frac{1-\cos\left(i \cdot \pi / w\right)}{2}, \tag{6}$$

as illustrated in Figure 19.

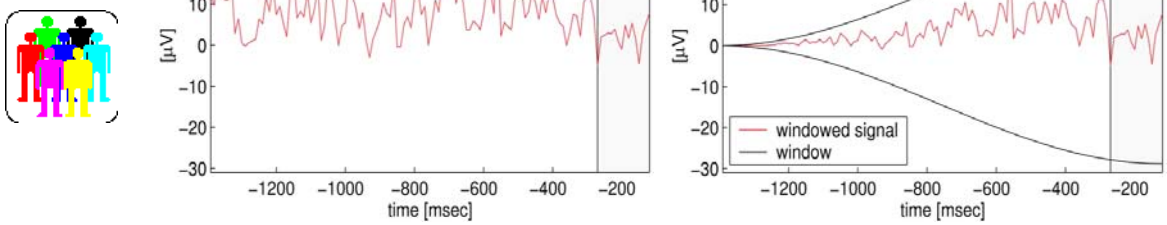


Figure 19: Windowing treatment of the pre-processing procedure Feature Selection. Data of a single EEG channel of an arbitrary sample extracted from continuous EEG (left) is multiplied by a cosine window (right) to emphasize the late information content.

A Fast Fourier Transformation (FFT) filtering technique is then applied to the windowed signal, producing complex valued Fourier coefficients (like those illustrated in the left part of Figure 20). Only the positive half of the frequency spectrum (the first 14 coefficients) is shown here as the magnitude value of the complex coefficient. We discard the baseline and all coefficients above a certain frequency, e.g. 5 Hz or even lower, such that only the δ-rhythm carrying the slow wave potential remains in the signal spectrum.

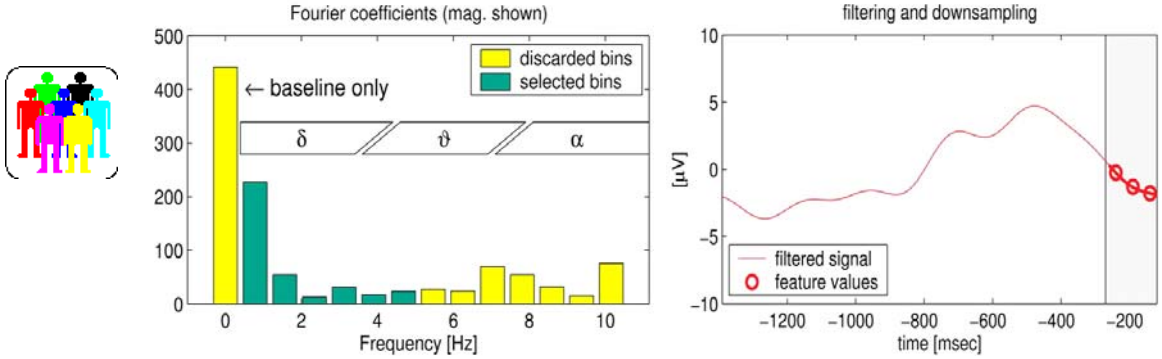


Figure 20: The FFT filtering technique performs low-pass filtering discarding the baseline bin (left) and the selection of 3 feature values from the tail of the signal (right), which has been transformed back into the time domain by the inverse FFT.

Transforming the low-pass frequency spectrum back into the time domain generates the smoothed signal, exemplarily depicted in the right part of Figure 20. The last 150 ms are then sub-sampled again at 20 Hz by calculating the means of consecutive non-overlapping intervals, each of five samples. This results in a number of feature components per sample and channel (here exemplified as three). Several parameters of the pre-processing procedure must be

determined with respect to the resulting classification accuracy in preceding offline or pseudo-online studies and must be set up before the experiment starts. These parameters are summarized in Table 6, where the reader can also view the appropriate default values, printed in italics in the right column.

Table 6: Appropreate (default) values for the parameters of the pre-processing procedure Feature Selection.

Parameter	Notation	Default values
Window width [data points]	`w`	32, 64, *128*, 256
Windowing function	`enumeration`	*Cosinus*, half-Gauss, Linear, None
Filter	`enumeration`	*Low-Pass*, High-Pass, Band-Pass, Notch
Low /High frequencies [Hz]	(f_{Low}, f_{High})	(0, 2.5), *(0, 3)*, (0.5, 4), (1, 5)
Trailer length [% of window width]	`tr`	7.5-22 (*12*)
Feature selection	`enumeration`	*Equidistant*, Linear
Number of features	`nF`	3, *4*, 5, 6

5. Classification

All feature values selected from the pre-processing, finally, are enqueued in a fixed manner, forming a high-dimensional feature vector for each training sample. For example, three features are selected from each sample and channel employed in a subset of the 40 most relevant EEG channels. The feature vector of each training sample may then be constructed from EEG acquired over both the left and right primary motor cortices and live in a 120-dimensional feature space. These vector data are to be classified by a discriminator with respect to the class label attached to the sample from which the vector's elements are extracted as features. Several regularized classification techniques have been implemented to be used in the BBCI system and tested in offline, pseudo-online and real online experiments. They included various linear classification approaches like Linear Programming Machines (LPM), linear Perceptron with weight decay, Linear Discriminant Analysis (LDA), linear Support Vector Machines (SVM) and Regularized Fisher's Discriminant [Blankertz et al., 2002]. Furthermore, some promising non-linear discriminative approaches, like Quadratic Discriminant Analysis (QDA) or Support Vector Machines (SVM) with Gaussian kernels have been applied to these feature vectors. Various non-linear classification methods are undoubtedly capable of handling more complex problems since they can adjust additional parameters. Yet, precisely due to their sophistication, they are more sensitive to noise which interferes with the data. Therefore, when applied to a simple, linear data set, they generally indicate a lower training error while producing a corresponding, significantly higher, generalization error. Please note also that noise is a principle concern in EEG data.

The phenomenon is exemplified in Figure 21. Two classes of data (circles and crosses) are distributed in Euclidean space with a trickle of positional noise such that a linear separation

yields a more general discrimination with respect to unseen data (illustrated in red). However, the non-linear curve performs perfectly when based on training data that tries to fit the positional noise of the data points located near the decision border – thus failing to interpolate the test data points.

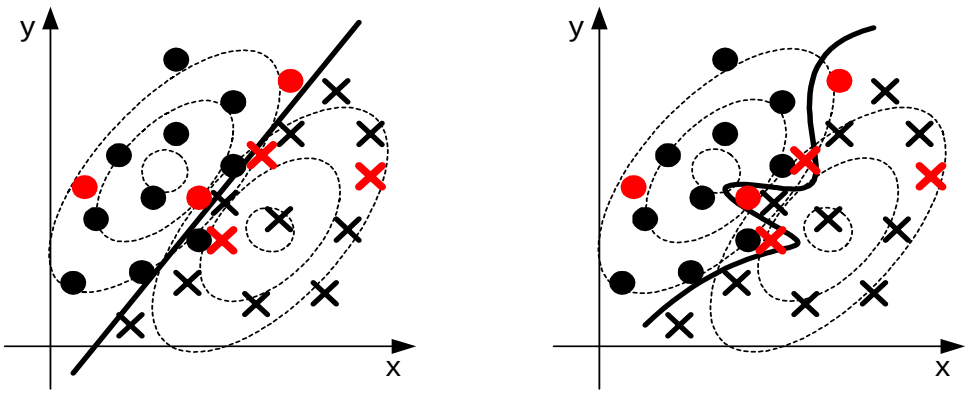


Figure 21: Example of linear (left) and non-linear (right) discriminative approaches applied to a toy classification problem. The training data set (black circles and crosses) can be separated with higher accuracy by the non-linear approach yielding a lower training error. However, it performs worse on unseen data (red circles and crosses) yielding a significantly higher test error. This phenomenon is also known as overfitting.

Nonlinear methods have not shown to produce greater classification accuracy, in the sense of cross-validation, than the best linear methods and often they performed even worse. This is in agreement with experiences collected from several BBCI experiments, in which ERP features of different classes of motor trials (e.g. left and right-hand finger movements) were distributed according to a simple process. It can be fairly assumed that the resulting data set is in essence linearly separable; however, due to the presence of environmental and some process noise, the overall data set appeared as being linearly non-separable. An ideal method will be able to recognize only those variations induced by the process, and ignore variations caused by noise. Yet linear classifiers are thus still the more appropriate as they are less sensitive to noise.

The pattern recognition theory says that the Fisher's Discriminant (FD) gives the classifier with minimum probability of misclassification for known normal distributions with equal covariance matrices [Vapnik, 1995]. As I will further explain in the following subsection, the classes of ERP features can be assumed to obey such distributions. Because the true distribution parameters are unknown, means and covariance matrices have to be estimated from training data. This is prone to errors since we have only a limited amount of training data at our disposal. To overcome this problem, it is common to regularize the estimation of the covariance matrix. In the mathematical programming approach of Mika et al. [Mika et al., 2001], the following quadratic optimization has to be solved in order to calculate the Regularized Fisher's Discriminant (RFD) w from data x_k and labels $y_k \in \{-1, 1\}$ and the data sample index k ($k = 1, \ldots K$):

$$\min_{w,b,\xi} \tfrac{1}{2}\|w\|_2^2 + \tfrac{C}{K}\|\xi\|_2^2 \qquad \text{subject to:} \qquad y_k\left(w^T x_k + b\right) = 1 - \xi_k \tag{7}$$

for $k = 1, \ldots, K$, where $\|\cdot\|_2$ denotes the ℓ_2-norm for which $\|w\|_2^2 = w^T w$ holds and ξ_k are slack variables. C is a hyper parameter that has to be chosen appropriately, say, by cross-validation

strategies. There is a more efficient way to calculate the RFD, but this formulation has the advantage that other useful variants can be derived from it [Mika et al., 2001], [Müller et al., 2001]. For example, using the ℓ_1-norm in the regularizing term enforces sparse discrimination vectors. The major advantage of the RFD-based classifier lies in its lower computational costs compared to the other above-mentioned methods; this allows the performance gain to be maximized.

5.1 Classifier analysis

The event-related potential (ERP) features are superpositions of task-related and many task-unrelated signal components, e.g. background auditory, visual, receptional, haptic, olfactory noise, or task-unrelated thoughts. The mean of the distribution across trials is the non-oscillatory task-related component – the ERP – ideally the same for all trials. The covariance matrix depends only on task-unrelated components. An analysis of these matrices calculated selectively for left-hand and right-hand movements showed that the distribution of the ERP features is indeed normal (see Figure 22).

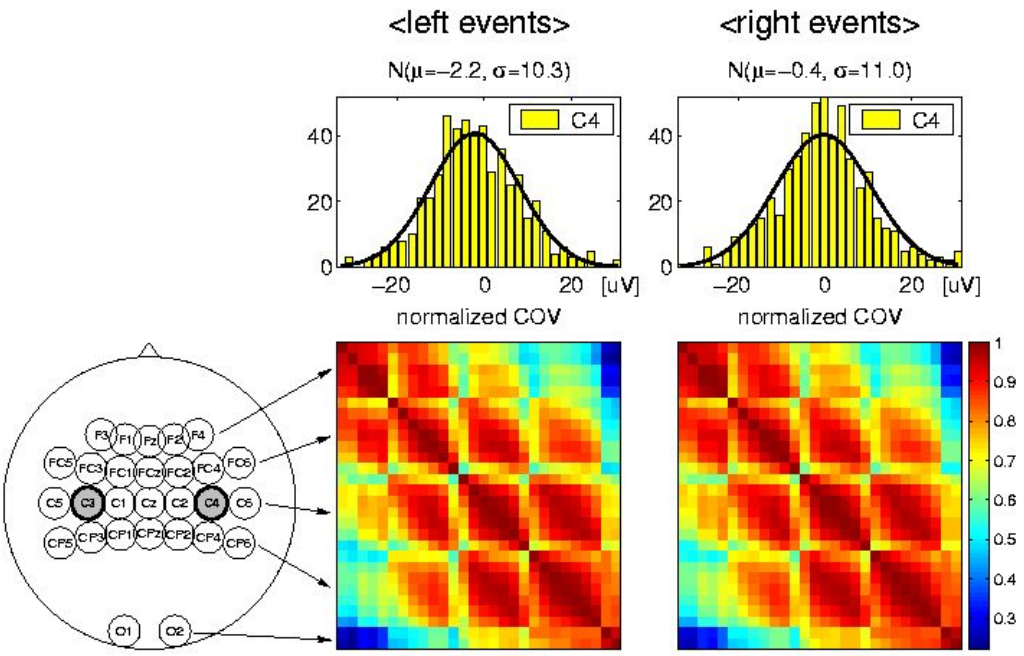


Figure 22: Distributions of the ERP features selected at electrode position `C4` for only left and only right events separately with a Gaussian curve fitted into the data (upper row) and the corresponding normalized covariance matrices (bottom row).

The covariance matrices are calculated only from a one-time slice of the ERP features, i.e. in regard to the SSP set, for $n = 1$ and for a fixed time point prior to a key stroke at t_d = -110 ms. Along each axis of the matrices, EEG channels are sampled in lines that go from the frontal to the occipital scalp, each line going from the left to right hemisphere, thereby causing the lattice structure of the covariance matrices. An important observation here is that

the covariance matrices of both classes look very similar. The histograms (see upper part of Figure 22) show the distribution of ERP features at channel C4 (located over the right motor cortex and thus corresponding to the activity of the left hand) at a fixed time point and superimposed by a fitted normal distribution. The normalized covariance matrices across channels for the two conditions (left vs. right-hand finger movement preparation) have only minor differences, most probably induced by noise. These minor differences are ignored by linear classification, whereas they are a potential concern for non-linear classifiers.

5.2 The Fisher's linear discriminant

This linear discrimination technique was introduced by Fisher [Fisher, 1936]. As pointed out in the beginning of this section, a low number of data points, which are of comparatively high dimensionality, confronts us with the "curse of dimensionality". This means that designing a good classifier quickly becomes more difficult along with the increasing dimensionality of the input space. The Fisher's Discriminant aims for an optimal linear dimensionality reduction. It is thus not strictly a discriminant by itself but it can be easily used to construct one [Bishop, 1995].

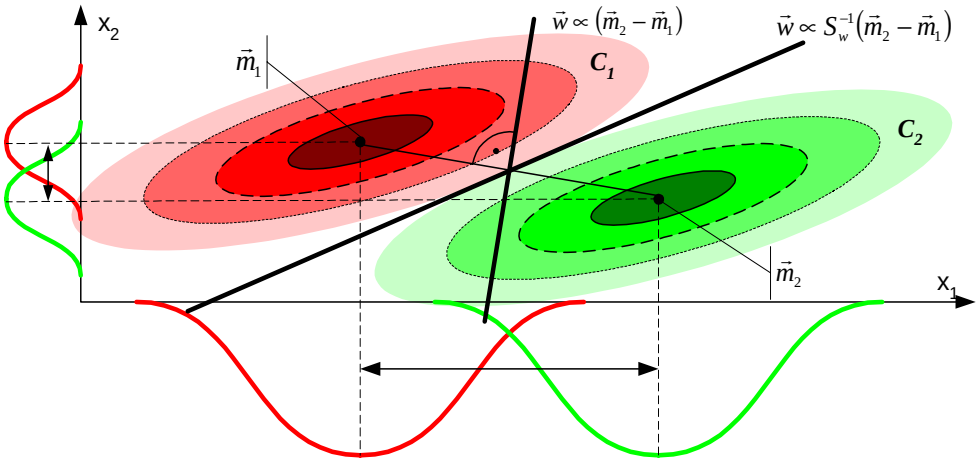


Figure 23: Schematic illustration of two classes separated by linear models with and without taking into account the within-class covariance matrix S_w . Distributions of the two classes of data (red and green) are shown with means and variances if projected onto the two coordinate axes. A simple linear discriminant does not yield satisfactory results, but the one optimizing the Fisher's information criterion (12).

A very simple approach of dimensionality reduction is to use a linear projection of the data onto a one-dimensional space. An input vector $\vec{x}$ will then be projected onto the value $y = \vec{w}^T \vec{x}$, where $\vec{w}$ is a vector of adjustable weight parameters. This gives the projection that maximizes the separation, i.e. the distance between the centers of mass of the two classes. For example, consider two classes C_1 and C_2 containing $|C_1|$ and $|C_2|$ data points respectively. The mean vectors of the two classes, i.e. their centers of mass, are then given by:

$$\vec{m}_1 = \left\langle \vec{x}_i \right\rangle_{x_i \in C_1} = \frac{1}{|C_1|} \sum_{i=1}^{|C_1|} \vec{x}_i \qquad \text{and} \qquad \vec{m}_2 = \left\langle \vec{x}_i \right\rangle_{x_i \in C_2} = \frac{1}{|C_2|} \sum_{i=1}^{|C_2|} \vec{x}_i \ . \tag{8}$$

We might define the separation of the classes, when projected onto the weight vector, as being the separation of the projected class means. Therefore $\vec{w}$ must be chosen to maximize

$$m_2 - m_1 = \vec{w}^T \left(\vec{m}_2 - \vec{m}_1 \right), \qquad \text{where} \qquad m_k = \vec{w}^T \vec{m}_k \quad \text{and} \quad k = 1,\ 2 \tag{9}$$

defines the class mean of the projected data from class C_k. The problem is then simply to find a weight vector such that

$$\vec{w} \propto \left(\vec{m}_2 - \vec{m}_1 \right)\Big|_{\vec{w}^T \vec{w}=1} . \tag{10}$$

However, as illustrated in Figure 23, projecting two classes, well-separable in their original two-dimensional space (x_1, x_2), onto the x_1-axis, gives a much higher separation of the projected class means than does a projection onto the x_2 axis. Therefore, the separation as defined in (10) is insufficient.

This difficulty stems from the substantial differences of the *within-class* spreads along the two axis directions. The solution presented by Fisher is to maximize the function that represents the difference between the projected class means, normalized by a measure of the *within-class* scatter along the direction of $\vec{w}$. The within-class covariance is given by:

$$\sigma_k^2 = \sum_{i=1}^{|C_k|} \left(w^T x_i \Big|_{x_i \in C_k} - m_k \right)^2 , \qquad \text{with} \qquad m_k = \vec{w}^T \vec{m}_k , \tag{11}$$

and the total within-class covariance for the entire data set may be simply defined as $\sigma_1^2 + \sigma_2^2$. The Fisher criterion could thus be defined as:

$$\begin{aligned} J(\vec{w}) &= \frac{\left(m_2 - m_1 \right)^2}{\sigma_1^2 + \sigma_2^2} = \frac{\left(\vec{w}^T \vec{m}_2 - \vec{w}^T \vec{m}_1 \right)^2}{\sigma_1^2 + \sigma_2^2} = \\ &= \frac{\vec{w}^T \left(\vec{m}_2 - \vec{m}_1 \right)\left(\vec{m}_2 - \vec{m}_1 \right)^T w}{\vec{w}^T \left(\sum_{i=1}^{|C_1|} \left(\vec{x}_i \Big|_{x_i \in C_1} - \vec{m}_1 \right)\left(\vec{x}_i \Big|_{x_i \in C_1} - \vec{m}_1 \right)^T \right) \vec{w} + \vec{w}^T \left(\sum_{i=1}^{|C_2|} \left(\vec{x}_i \Big|_{x_i \in C_2} - \vec{m}_2 \right)\left(\vec{x}_i \Big|_{x_i \in C_2} - \vec{m}_2 \right)^T \right) \vec{w}} = \\ &= \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}} \end{aligned} \tag{12}$$

where S_B defines the *between-class* covariance matrix, and S_W is the total *within-class* covariance matrix. This can be easily deduced from (12) and is given by:

$$\begin{aligned} S_B &= \left(\vec{m}_2 - \vec{m}_1 \right)\left(\vec{m}_2 - \vec{m}_1 \right)^T \\ S_W &= \sum_{i=1}^{|C_1|} \left(\vec{x}_i \Big|_{x_i \in C_1} - \vec{m}_1 \right)\left(\vec{x}_i \Big|_{x_i \in C_1} - \vec{m}_1 \right)^T + \sum_{i=1}^{|C_2|} \left(\vec{x}_i \Big|_{x_i \in C_2} - \vec{m}_2 \right)\left(\vec{x}_i \Big|_{x_i \in C_2} - \vec{m}_2 \right)^T . \end{aligned} \tag{13}$$

The weight vector must be chosen such as to maximize the Fisher criterion. The problem is thus reduced to a simple quadratic optimization problem. When differentiating the Fisher criterion with respect to $\vec{w}$, it becomes clear that it is maximized when:

$$\left(\vec{w}^T S_B \vec{w} \right) S_W \vec{w} = \left(\vec{w}^T S_W \vec{w} \right) S_B \vec{w} . \tag{14}$$

Since $S_B \vec{w}$ is always in the direction of $\left(\vec{m}_2 - \vec{m}_1 \right)$, deduced from (13), and since we do not care about the magnitude of $\vec{w}$ but only about its direction, we may drop all scalar factors, multiply both sides by S_W^{-1} and obtain, consequently, the weight vector to be calculated as:

$$\vec{w} \propto S_W^{-1} \left(\vec{m}_2 - \vec{m}_1 \right) . \tag{15}$$

This is what is ultimately known as the Fisher's Linear Discriminant. However, strictly speaking it is not a discriminant but rather a specific choice for a direction of the data projection down to one dimension. Please note that if the *within-class* covariance is isotropic, the weight vector $\vec{w}$ becomes proportional to the difference of the class means, as discussed above.

Additionally, we can choose the bias y_0 such that a new point $\vec{x}$ can be classified as belonging to the correct class, i.e. either C_1 or C_2 according to:

$$\vec{x} \in \begin{cases} C_1 \Leftrightarrow \vec{w}^T \vec{x} \geq y_0 \\ C_2 \Leftrightarrow \vec{w}^T \vec{x} < y_0 \end{cases}. \tag{16}$$

For example, the bias parameter y_0 can be chosen as the intersection point of the Gaussian distributions; this parameter is estimated from the data points and projected onto the weight vector of the discriminant $\vec{w}$. To summarize, the Fisher's Discriminant searches for a separating hyperplane that will subdivide the feature space into two classes. Thereby it optimizes its model parameters in two ways: (*i)* it maximizes the distance between the centers of mass of the two classes, i.e. the so called *between-class* variance; and (*ii)* it minimizes the covariance of the data within each class, i.e. the so called *within-class* covariance. The principal separation procedure used by (R)FD and its goals are illustrated in Figure 24.

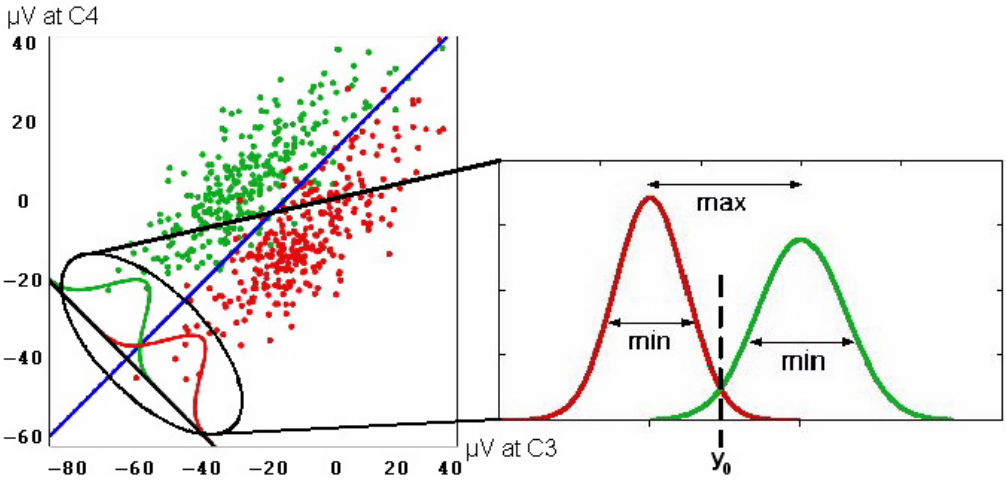


Figure 24: Principal separation aims of the Regularized Fisher's Discriminant (RFD) exemplified on a two-class problem with 2D-data. The RFD is searching for a separating hyperplane that maximizes the distance between the means of distributions of the two classes and minimizes the variances within each class when the data is projected onto the weight vector.

5.3 Classification results

As outlined earlier in Section 3, to enable the classifier training, i.e. to calculate the user model parameters, the user is instructed to execute or imagine the task accomplishment repeatedly, according to the experimental setup and the introduced task. EEG data is continuously acquired during the training session, and training samples are then extracted from it (see Section 3.2). These samples are pre-processed with a procedure that has been specified by the experiment conductor. Pre-processing can also yield the parameters of the pre-processing module of the user model (see Figure 14) to be applied to the data in later online sessions. An optimal classifier can then be calculated from a $(1-1/n)\cdot 100\%$ subset of the available feature

vectors. This classifier is then tested on the remaining $1/n \cdot 100\%$ subset which contains data that the classifier is not based on, i.e. data unseen by the classifier. The complete procedure is then repeated n times for all non-overlapping test sets, resulting in n slightly different classifiers. This n-fold cross-validation (sometimes also mentioned as "*X-Validation*") procedure is illustrated in Figure 25.

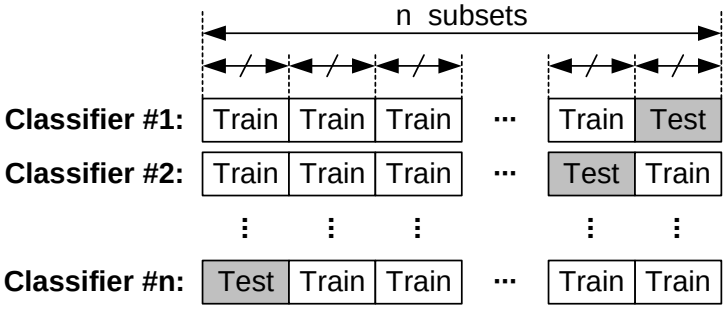


Figure 25: n-fold cross-validation procedure. The entire dataset is divided into n subsets, n-1 of which are used always for training but the remaining one for testing yielding n different classifiers. Test errors are calculated from the application of each classifier to the remaining subset containing only unseen data. The cross-validation error is then obtained as an average over all test errors.

In the m×n-fold Cross-Validation procedure employed for testing the classifier, the data set is repeatedly rehashed in random order m times, resulting in m×n training and test errors. The cross-validation error is then obtained by averaging over all test errors. This yields a good measure for the quality of the classifier model applied to the given data. An averaged test error, essentially higher than the averaged training error, would indicate either that the classifier model is too complex with respect to the presented data or that the data set is too small. This results in a poor interpolation ability of the classifier, due to poor generalization ability and an essentially high risk of over-training (which we have also referred to as overfitting). Please note that, to produce fair results, the rehashing strategy must be based on the set of events and not on the set of single samples, i.e. the rehashing must change the "position" of the events within the training data such that samples that have been extracted from the same event marker, i.e. belonging together, remain in the same either training or test set after rehashing.

Notably, the test errors of the cross-validation procedure depend on the choice of the delay time t_d in the pre-processing procedure. Obviously, the classification is ambiguous for large values of t_d, and increasingly easier as $t_d \to 0$ since the amount of information about the action grows as the event (executed or imagined movement) approaches. Figure 26 shows the cross-validation error obtained by the 10×10-fold Cross-Validation procedure. This error is estimated for the classification of single-trials as a function of the delay time t_d for a single subject performing in a self-paced experiment with 30 taps per minute (upper plot, slow pace) and 120 taps per minute (lower plot, fast pace).

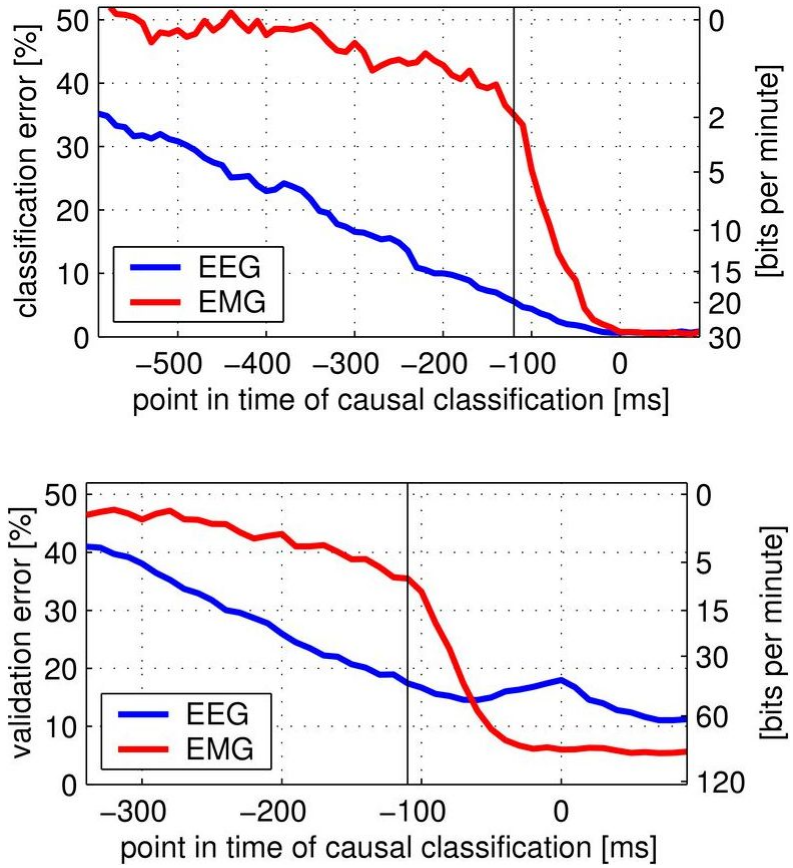


Figure 26: Cross-validation errors of the classification in a slow pace (top) and a fast pace (bottom) experiment based on EEG features (blue curves) and EMG activity (red curves) estimated by a 10×10-fold cross-validation procedure. The error is plotted as a function of the time point of causal classification, i.e. the t_d parameter of the SSP.

The right ordinate in both plots enumerates the theoretical information transfer rate, in bits per minute (bpm) that can be extracted from the classification results. The amount of information I as a function of the accuracy p of the system is defined by Shannon as:

$$I(p) := \log_2 N + p \cdot \log_2 p + (1-p)\log_2\left(1 - p/(N-1)\right), \tag{17}$$

where N denotes the number of classes (here $N = 2$). The classification based on EMG activity (red curve) that reflects muscle exertion in the forearms is performed by analyzing the oscillatory behavior of data in the corresponding channels, i.e. `EMGl` and `EMGr`. Since the amplitude of the oscillations rises when muscular actions take place, it is sufficient to calculate the variance of the data points that appear in the sample window. This analysis yields the features plugged into the classifier. However, the EEG-based approach yields significantly superior classification results already 120 ms prior to the actual movement execution (indicated in the Figure by a vertical cut-off line) and retains this performance level until the time stamp of the key tap in the slow movements experiment. From the view of neurophysiology, this is explained in that the decision to lateralize the movement begins in the brain firstly, followed by the preparation of the cortical neurons and the emission of the command down to the spinal

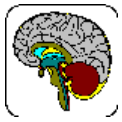
cord, the peripheral nerves and the effector muscles; the entire process takes at least 70-100 ms.

6. Error-signal Detection

Von Natur gibt es weder Gutes noch Böses — den Unterschied hat nur die menschliche Meinung gemacht.
Sextus Empiricus (~ 200 A.D.)
Das am notwendigsten zu erlernen sei das schlechte zu verlernen.
Diogenes Laeritus (~ 300 A.D..)

It is a well-known finding in human psychophysics that a subject's recognition of having committed a response error is accompanied by specific EEG variations that can be easily observed in averaged ERPs. Similar variations can also be detected in users observing an unusual situation, e.g. an unexpected feedback animation. Blankertz et al. [Blankertz et al., 2002a] present a pattern recognition approach that allows for a robust single-trial detection of this error potential from multi-channel EEG signals. An elegant approach to overcome the problem of low classification accuracy is to base the response checking mechanism on the subject's brain signals themselves. In this way, those persons benefit most who can otherwise attain only modest BCI control due to a considerable portion of classification errors.

An essential question concerning error potentials in BCI applications is whether the user recognizes the error made by the BCI system's signal recognition module as a "committed response error". In BBCI feedback-based application sessions, the user is provided with the reactions of the system; currently, this is visual or auditory information based on the system's decision about user action or intention. However, as outlined in the previous section, a classification is never perfect, such that the user is sometimes given feedback information she/he did not expect. The user then develops an annoyance about the mismatch, which in turn generates a special type of event related potential (ERP), the so-called error potential.



An ERP after an error trial is characterized by two components: a negative wave (N_E) with a fronto-central maximum, and a subsequent positive peak (P_E) with a centro-parietal maximum. N_E seems to reflect some kind of comparison process initiated as it is present in most trials (also in correct ones). P_E seems to indicate the brain's reaction to having recognized that the subject's action was erroneous. Figure 27 shows averaged miss-minus-hit EEG traces at four selected electrodes along the vertex together with the corresponding scalp topographies.

The N_E peak observed during situation assessment is a special case of the *N100* potential, while the P_E peak is a special case of the *P300* potential; both indicate their timing of occurrence. For example, *N100* reaches its maximum negativation about 100 ms after the user has decided to perform the fault action, while *P300* reaches its maximum positivation at about 300 ms, also relative to the timestamp of the decision. However, the curves shown in Figure 27 illustrate the maximum of the N_E peak at about 20 ms and that of the P_E peak correspondingly at about 230 ms relative to the action timestamp. This is evident due to the setup of the "d2-test" experiment, which is described briefly in the next subsection.

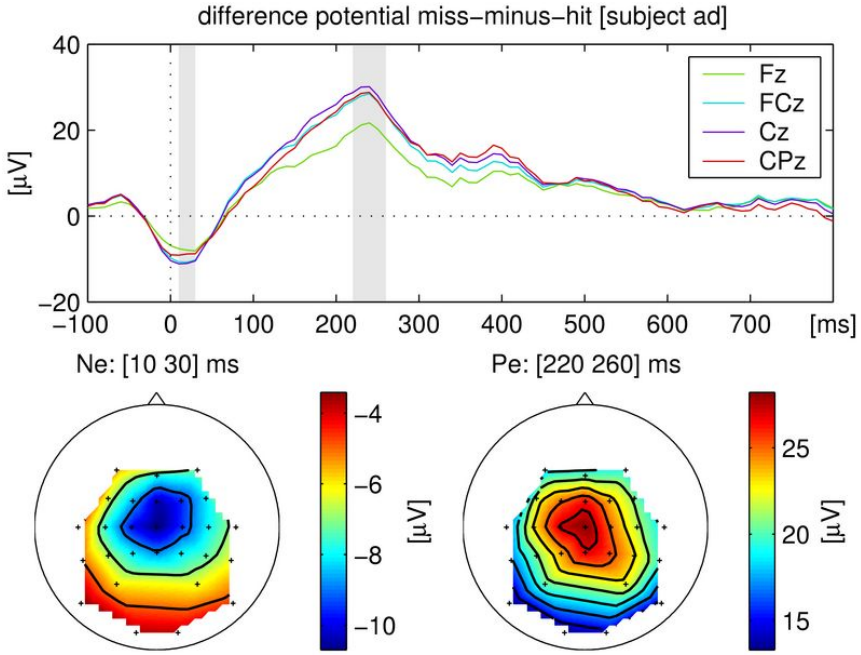


Figure 27: Averaged error potentials (top) illustrate a negativation (*Ne*) followed by a strong positivation (*Pe*) after user's erroneous decisions in the d2-test experiment. The corresponding scalp topographies (bottom row) illustrate the fronto-central maximum of the error negativity (left) and the centro-parietal maximum of the error positivity.

6.1.a The "d2-test" experimental setup

The user is provided with a query on the screen; this query can be either a target or a non-target. The queries resemble each other to the extent that it is difficult for the user to recognize at the first glance if she/he is provided with the target or the non-target query. Ascertaining the right query requires concentration and time in order to process the visual stimuli. The user has to respond as fast as possible to target stimuli by pressing certain keyboard key with the right hand, e.g. '`J`', and to non-targets, by pressing another key with the left hand, e.g. '`F`'. The user is provided with the green "OK" or the red "Fault" message right after the key press. The combined information about the stimuli presented to the user and the pressed key then yields the knowledge about the presence of the error-potential in the user's EEG during the trial.

The stimuli consist of a letter, either "`b`" or "`d`", and at most two horizontal bars above and two below the letter. Targets are described to the user as containing the letter "`d`" and exactly two bars, no matter where they occur. All remaining symbols are defined as non-targets, i.e. those containing the letter "`b`" with an arbitrary number of bars, or the letter "`d`" with a number of bars differing from two. Figure 28 summarizes all possible target and non-target symbols. Thus, early error potential can be explained by the effort to initiate an action, i.e. the correct key tap, as fast as possible by trying to predict just at the first glance the affiliation of the query to the class of targets or to that of non-targets. In fault trials which were performed on that motivation user then recognizes a fault action quite early, because the fault comes from the user's decision, not from feedback. The user is trying to cancel it, but cannot stop the emitted "*go*" signal any more since the "*veto*" signal is emitted too late and cannot catch up with and eliminate the previously and erroneously emitted "*go*" signal.

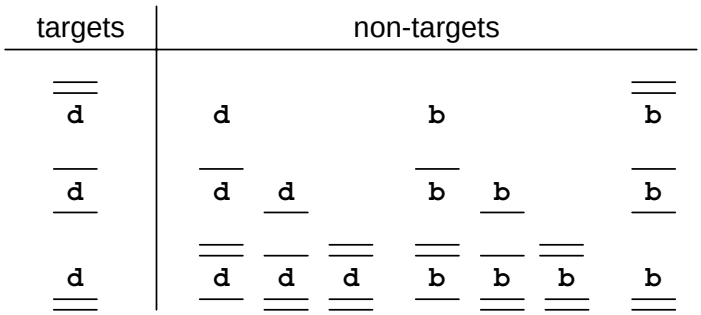


Figure 28: Targets and non-targets in a "d2-test" experiment. A target stimulus is composed of the letter d and exactly two horizontal bars placed above or below it. All remaining symbols are non-targets.

6.2 Additional benefits from errors

Other work [Schalk et al., 2000] reports the presence of an error related ERP in trials where the user is confronted with a mismatched feedback response. If a certain control command is recognized by the system, then the feedback response (e.g. the animation) can be flipped through a binary decision. It then corresponds to the opposite one and thus provides the user with an unexpected situation. This procedure employed during the training session provides the learning machine with fault trials containing the samples with the error-related EEG. The generation of an error-related potential by the user during the application session and the recognition of such a potential by the system lets the user deriving its benefits, since it is now possible to correct the last feedback response. If the user's decision is not of a binary nature, the last animation action can be taken back.

To evaluate how error detection could potentially improve BCI transmission rates, we assume a BCI system that provides 85% accuracy ($p = 0.85$) and calculate the amount of information that can be gathered from a two-class decision experiment ($N = 2$) using *Shannon*'s information criterion (17). This yields $I(0.85) = 0.39$ bits per selection. Please note that this is only a theoretical value used to measure BCI performance. Actually achieving this information transmission rate would require some specific and efficient coding of the information by the BCI user – a task humans may not be able to carry out. However, a system that (*i)* detects errors after each decision, and (*ii)* analyzes the user reactions to the feedback stimulation, and (*iii)* corrects the feedback animation according to the recognized error potential with 20% of false-negatives ($FN = 0.2$) and 3% of false-positives ($FP = 0.03$), can improve the information transmission rate by more than 75%, attaining 0.69 bits per selection. The accuracy of the improved system $\hat{p}(p, FP, FN)$ can then be calculated by:

$$\hat{p}(p, FP, FN) := p \cdot (1 - FP) + (1 - p) \cdot (1 - FN) \tag{18}$$

resulting in an improved accuracy of 94%.

The plot in Figure 29 sows the theoretical information rate I in a two-class experiment as a function of the accuracy p of the pure BCI system, with and without the error recognition and correction procedure, and working with an assumed rate of 20% of false-negatives and 3% of false positives. Obviously, the gain diminishes, the higher the accuracy is. Please note also that with the assumed parameters, an error correction approach is useful only as long as the pure BCI accuracy is below 96%.

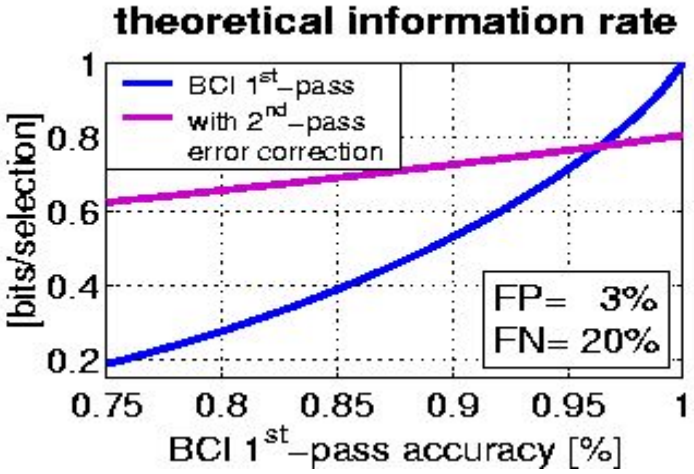


Figure 29: Improvement of the theoretical information rate *I* of the BCI system by the error correction strategy as a function of the pure BCI accuracy *p*. This plot assumes an error detection module working with 20% of false-negatives and 3% of false-positives.

6.3 Training with errors

The error detection and a consecutive error correction approach metamorphoses the already powerful BBCI system into an even more efficient and flexible machine. This constitutes its nature as a doubly learning system with an automatic response checking mechanism in the second pass. Let us now go back to Figure 13 (page 39), which described a simple training session procedure of the BBCI, and extend this training model by the noisy feedback. The aim here is to kill two birds with one stone, namely (*i)* to perform the basic training procedure for the user model and (*ii)* to learn how the user makes mistakes or, more precisely, to learn what happens when the BBCI makes errors. The former procedure can be performed in the first phase of user training yielding the user model as described previously. In the second pass when the user model is present, it is then possible to perform the latter procedure yielding the user error model.

The machine training process of setting up the user model of the BBCI system with an error detection module is illustrated in Figure 30. The user is given a task which she/he has to perform during training; the task may be coupled with the feedback and thus provide the user with stimuli. In this case, information about present stimuli is supplied to the BBCI system. While repeatedly performing the task, the user generates brain signals that are acquired by the main acquisition module running on the recorder PC. This information is then sent to the BBCI system, which, in addition, has its signal recognition module running. This module can be set up as described earlier during the first training session, i.e. based on the information provided by the task markers. When it is set up, it can also make the decision about the user's intentions. Consequently, the feedback module receives a command from the BBCI system and executes it. However, sometimes and with a certain probability it will execute the exact opposite (flip-side) command. This results in the user being confronted with a wrong animation or sound. The purpose of this action is to emit errors in a controlled mode, to which the user then generates an error potential. The marker about a change in the decision is then placed into the data such that the BBCI can recognize it and assume that some samples about the user's frustration, produced by the user's error generation module, are located in the data. The error recognition

model is then set up based on these samples. They are set into opposition to those samples that were extracted from the data while a correct feedback signal was given to the user; this allows recognizing the user's apperception of being provided a wrong feedback.

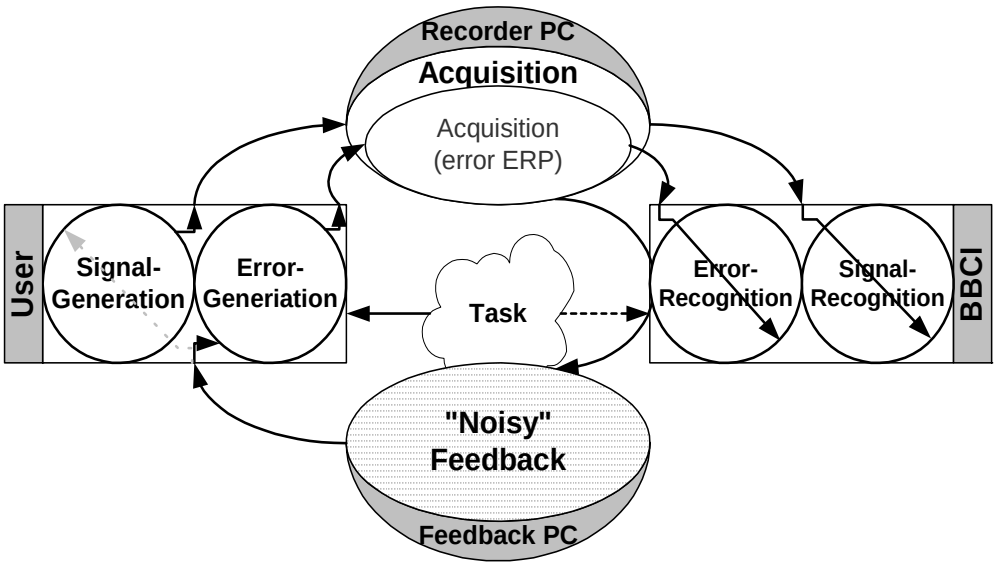


Figure 30: Machine training session of the BBCI system with automatic error recognition and feedback module with controlled noise. During the second pass of the training session the user is provided with a mismatched stimulus produced by the "Noisy" feedback module (bottom). She/He generates an error signal upon which the BBCI's error recognition model is setup.

Please note that during this second pass of training, the user's signal generation model is still adjustable such that the user provided the feedback reacts differently compared to the first training session which was not based on real feedback, although it was based on stimulation.

In the following application session, the BBCI system develops to a doubly learning system. Please compare now the Figure 15 (page 41), which describes the standard application session of the BBCI. In contrast, Figure 31 illustrates the process of applying the user model with an error detection module for controlling the feedback application.

The task now has the aim of gaining as much control over the feedback application as possible while the user is provided with the true feedback stimuli (i.e. to the best of BBCI's ability). The user generates brain signals that, acquired from the acquisition machine, are redirected to the BBCI's signal recognition module. The BBCI decides about the control command, and the feedback application provides the user with the corresponding animation or sound. After a control command has been sent out to the feedback module, the BBCI falls into the error recognition mode, thus monitoring the user reaction to the change in the feedback. If the feedback was to provide the user with the correct stimulus, the user would not generate any "surprising" activity, which, consequently, will most likely not be recognized as an error signal and end up in the confirmation of the last action of the feedback module. If, on the other hand, the user is confronted with unanticipated animation or auditory stimuli of the feedback, an error pattern will be generated and subsequently classified by the BBCI's error recognition module as an error signal. The BBCI then commands the feedback application to flip the last action if a two-class decision was assumed or to cancel the last, probably incorrectly recognized, decision in a multi-class setting.

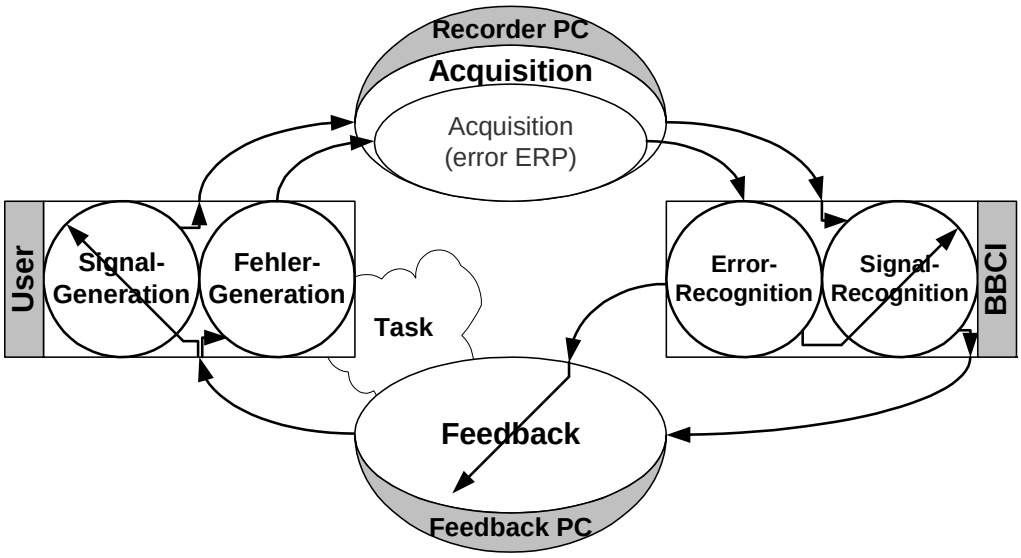


Figure 31: Application session of the BBCI system with automatic error recognition and a subsequent correction of the feedback stimulus. The feedback application is controlled according to the application of the user model to the data. However, on being provided a mismatched feedback stimulus the user generates an error potential that can be recognized by the BBCI's error recognition module. The feedback application then corrects its animation, and the BBCI's signal recognition module can adjust its user model.

It is also conceivable to incorporate the consequences of an ongoing error detection in order to re-train or to perform a fine-tuning of the BBCI's signal recognition module; however, this cannot be addressed in the scope of this thesis and requires further investigation.

7. Online Application with Feedback

In this last section of the chapter, we will discuss the final stage of the BBCI's information cycle. Closing the loop started by user connected to the acquisition machine, data is here passed to the trainer or the applier module, leading us to the feedback modules. These feedback modules are autonomous and interactive applications that aim to connect the BBCI output to the user. They can be run on a separate computer and receive their control command from the BBCI applier through the network. Feedback applications are designed to rely on simple control strategies which, expressible by a small command set, should provide the user with a feeling of being inside the simulation and acting naturally. Moreover, the underlying interactive feedback application should be intuitive and simple to understand. However, at the same time, it should avoid rapid animation and high changes in contrast in order to prevent or at least minimize the spoiling of data affected by artifacts, e.g. brisk eye, head or body movements. Of particular importance for fast-pacing of control commands is a "natural mapping" of the action required in the virtual reality or gaming scenario to the "action space" of the human operator, e.g. the BBCI user. This action space is obviously coded in egocentric coordinates. To this end, the on-screen environmental perspective must continuously represent the viewing direction of the human operator, so that, for example, the option to turn right could be selected by the intention to move the right hand and vice versa.

Most feedback applications, at least those based on command control incorporate a temporal queue of control commands received from the BBCI applier. The temporal queue is implemented in a cyclic manner, such that its function is similar to a LIFO stack. However, the data becomes overwritten within a certain time period defined by the length of the stack and the rate of data arrival. The receiving procedure is occupied exclusively with retrieving the command block and with inserting it appropriately into the queue. This procedure then emits a local broadcasting message indicating the update of the queue. Instances programmed to process this kind of messages can then begin analyzing the semantics of the latter part of the control command queue upon receiving such an update message. The *Queue Checker* is the main instance that waits for the update message. It examines a certain number of previously acquired control commands in order to determine whether a stable control signal is present for at least the period of the Command Activation Term (CAT). The semantics of the signal's stationarity can be determined by implementing the *Queue Checker*, which can check, for example, the concordance of the command affiliation to a particular class of interest. In addition, it can check if the estimated affiliations (their fuzzy values) are above a certain threshold, i.e. the Command Activation Threshold (CATh). If a stationary signal is present, the last control command inserted into the queue (the stationary signal's master command) is emitted to the execution instance that controls the on-screen animation of most applications.

Usually the user's brain signal remains stable for longer than a single CAT, a situation that initiates the repeated emission of several identical control commands. To overcome this ambiguity, a control command emitted to perform changes in the animation of the feedback application discourages the emission of further control commands for a certain time. This interval, called the Command Relaxation Term (CRT), and its function is comparable to firing properties of biological neurons that become "tired" after deploying a spike train and need a certain relaxation period to restore their firing capabilities. For technical reasons constrained by the implementation, the CRT must be at least as long as the CAT. All control commands received from the BBCI applier during the CRT continue to be inserted into the temporal queue; however, in this case the *Queue Checker* ignores incoming update messages. As a result, the role played by the acquisition procedure is not affected by the command emission, but vice versa.

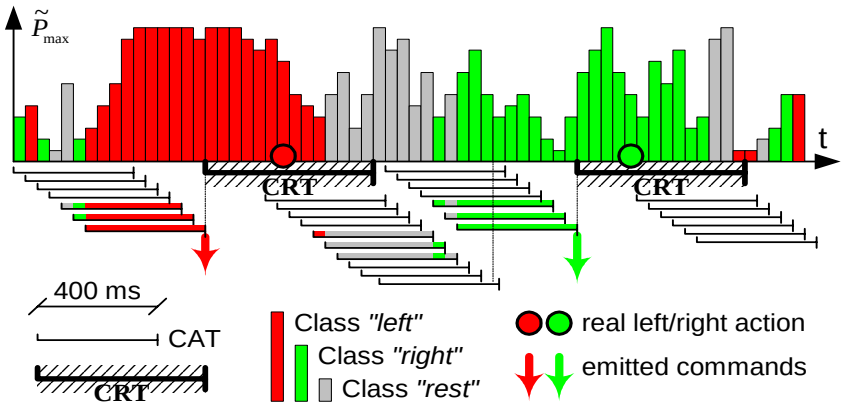


Figure 32: Procedure of command emission based on the temporal command queue. Three types of command blocks designated with their strength (fuzzy value) are acquired in the queue. When a stable signal is present for the CAT duration (here 10 periods) a control command is emitted. Further investigation of the queue content is then blocked for the CRT duration (here 14 periods).

This procedure is exemplified in Figure 32 for a two-class (left vs. right finger movements) self-paced experiment; a third class of "resting" is introduced in order to encode the information for some intermediate state where the user is not planning or intending to emit any of the two control commands. The last stage of the BBCI applier (the combiner) yields the class index, denoted here by colored bars and a corresponding fuzzy value. This provides a measure of certainty for the class label that is indicated by the amplitude of each bar.

The BBCI applier yields a single command block consisting of a class label and a corresponding fuzzy value each time the processing of a raw EEG data block is completed, i.e. each 40 ms. Consequently the command blocks are inserted into the queue at a frequency of 25 Hz. The fuzzy value is calculated according to the strategy defined by the combiner module of the BBCI applier. Most often, the "Winner Takes All" (WTA) strategy, the most probable guess, is incorporated ($\tilde{P}_{\max} = \max_{i \in Class} \tilde{P}_i$, where index *i* runs over all possible classes of interest). Exemplarily, the CAT is set to 10 periods (400 ms), while the CRT is 4 periods longer (560 ms). The emission of a command to the animation module of the feedback application is illustrated here as a bold arrow that has the color of the class label. The event marker of a real finger movement contained in the data is illustrated as a circle; its fill color also corresponds to class information. Notably, the emission of the control command is initiated at some arbitrary, but in most cases earlier, time point that is relative to the performance of the action; thus, the "arrows" often precede the "circles".

This flexible setup of command emission allows for the individual adjustment of parameters such as CAT and CRT, which strongly rely (*i)* on the experimental setup, (*ii)* on the individual capabilities of the user and the user's brain signal properties, and (*iii)* on the control strategy currently selected by the user for gaining control over the feedback application. A CAT parameter set to large values helps avoiding false positively emitted commands. Since the classification may not be correct for every single command block, a long CAT prevents the repetitive emission of many short control commands with ambiguous class labels or even trains of these control commands. In contrast, the CAT parameter cannot be set to any arbitrary long terms because the duration of a commonly generated stable signal is in most cases limited by neurophysiological constraints. The negativation of the signal the classifier is looking for cannot continue for an arbitrary long term. In experiments with real movements it has been observed that the negativation usually ends once an action takes place. A short CAT would allow a fast emission of control commands, i.e. a reasonable term before the real movement is executed. An intra-individually adjusted CRT prevents erroneous, and respectively allows volitional, successive emissions of the latter command. Exemplarily, if a cursor control task is employed as the feedback application that is to move the cursor to a predefined and fixed position (target), the user may need to relocate the cursor as fast as possible from position `A` to target `B`. As this target may be far away, e.g. several steps from `A`, subsequent command emissions will be needed. For example, in the Brain-Pong feedback application, which can also employ a command-based control strategy, the operator often has to reposition the racket quickly from one side of the window to the other in order to hit the ball that is previously bounced off with a large angle. Therefore, in particular for this application, when used in conjunction with a command-based control strategy small CRT settings are preferable.

A practice-oriented reader should, however, notice that these parameters are also highly subject-specific. Thus, they should be initially set to values calculated from the results of the application of a "trained" classifier to the entire stream of the training data, resulting in several periods of a stable signal with different durations. Initially, the CAT_0 should be set to the median length of the stable signal duration that contains a marker of the identical action class. While the initial relaxation term (CRT_0) should be set to a value larger than CAT_0 by twice the amount of the standard deviation of the distribution of stable signals' durations. These values are then to be continuously adjusted according to the user's demand until they become settled.

7.1 Feedback scenario “Jumping Cross”

In the first BBCI feedback-based experiments, it was of major importance for the user to “*feel*” a direct connection between her/his own intentions and the feedback application controlled by the system. It was originally unclear whether, and if so how much, the operator’s brain signals would change from the first training session, which is completely without any feedback, to the application session accompanied by the feedback. Aside from the increasing visual attention and concentration, the data from feedback-based application sessions might become much more affected by artifacts resulting from the user reactions to animation. In particular, we expected the system to be able to recognize the user’s intentions of performing an action a moment before the execution of the intrinsic action. The system would then be able to indicate that recognition to the user via the feedback application. Consequently, the following interesting question arises: “How would a subject react when confronted with her/his own future action (immediate future, yet future nonetheless)?”

Therefore, a very simple visual feedback application was implemented to simulate the feeling of being confronted with one’s own intentions. The user faces a full-screen window with a neutral background color, e.g. light gray. That window contains a thin static fixation cross in its center. The user is instructed to steer her/his ocular attention to the center of this cross, proceeding as smoothly as possible without following the animation with the eyes (the latter to minimize artifacts induced by the electrical dipole “eye” controlled by the tracking muscles). Two large, pentagonal target fields are located in the upper corners of the screen – a dark red one in the left corner, and a dark green one in the right corner – indicating left and right-hand movements respectively. These target fields can change colors, e.g. become highlighted or even take on an arbitrary color at certain events. These events can result from response markers placed in the data by the user’s action such as tapping on a keyboard, or even from the cursor as it moves into the corresponding field. The cursor is displayed here as a small bold cross which can be moved by the user’s intentions over the entire screen. The ordinate axis of the screen’s coordinate system reflects the normalized decision of the detection classifier. Lower values indicate rest, while higher values indicate the fuzzy value of the detection classifier’s certainty about processing an action sample or a sample shortly preceding the action. In this way, the cursor, i.e. the cross, “jumps” into the upper part of the screen upon upcoming action and remains in the lower part of the screen when the user is relaxing. The abscissa guides the natural mapping of the determination classifier result, indicating the lateralization of the upcoming action.

The cursor trails a tail of points depicted as small balls (their number and filling color is defined in the initialization file). The tail balls are connected by thin lines and become smaller the more the corresponding data sample is distant in time. The time interval between history balls can also be defined in the initialization file. To do so, the complete history tail of cross coordinates must be saved in an array of the length, defined by the product of the number of balls and the time interval between subsequent balls, divided by the control command rate.

Consequently, in the ideal scenario the cross should fidget at the bottom of the screen and “jump” to the upper-right or upper-left corners upon oncoming right or left actions respectively. As a result, at the end of each trial the cursor will be located in one of the target fields. The entire simulation will then freeze for a certain period of time, indicating the end of a trial. For example, a trial is finalized when a key press marker arrives in the data. Please refer to Figure 33 for an illustration of the feedback scenario “Jumping Cross”, depicting the relaxing state in the upper part (both target fields are dark and the cursor is fidgeting in the lower half of the screen). Depending on the user’s intention, the state may change to the one of the two action states shown in the bottom row (either the left or right target field contain the cursor and are highlighted).

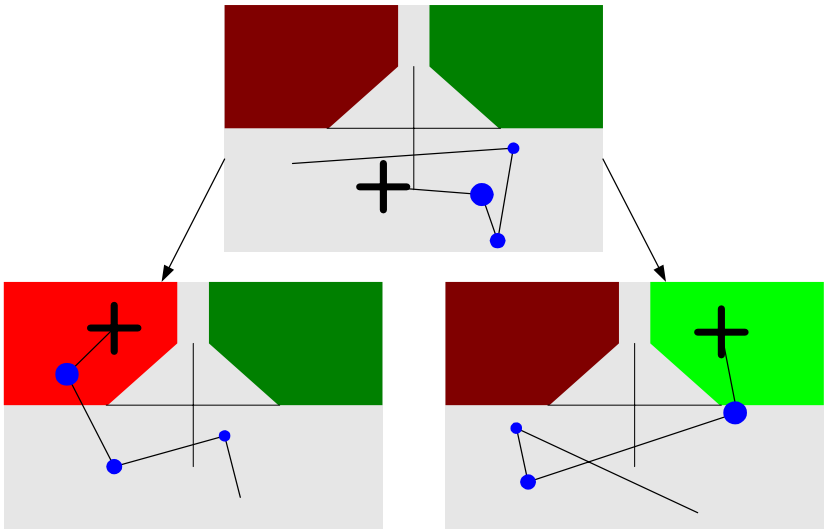


Figure 33: Three states of the feedback scenario “Jumping Cross”. The cursor is located in the lower part of the screen, when the user rests (top); it “jumps” to the upper-left or upper-right target field (bottom row) upon the user’s intention to move the corresponding hand. The target field currently containing the cursor becomes highlighted.

7.1.a Self-paced experiments

In self-paced experiments actual movements of healthy subjects are investigated. The movement laterality as well as the time stamp of the action is therefore provided to the BBCI system. The feedback application is consequently able to perceive this information via event markers placed into the data by the user, who performs actual key tapping on a computer keyboard. The user was given three different specifications of the feedback control in order to clearly demonstrate the evidence of the BBCI approach.

- Self-paced and self-lateralized movement with effect observation. Here the user may emit any of the two actions at any time and is able to observe the feedback animation. However, in this experiment the user could wait for the cursor to jump into either the left or right target field, in order to then quickly press the corresponding key to correctly finalize the trial by freezing the simulation. Thus, the user could possibly “cheat” in a way that cannot be monitored by the experiment.

- Self-paced and self-lateralized movements without effect observation. Here, the feedback monitor is turned away from the user, such that it is still observable to the experiment conductor who is now required to keep a closer eye on the proceedings. The conductor is required to closely observe the monitor and the user’s hand movements in order to ensure a correct correspondence between hand movements and the position of the cursor. The cursor should be located in the target field that correctly corresponds to the sidedness of the performing hand movement after the simulation freezes. The majority of correctly finalized trials have shown a successful correspondence and confirmed the evidence of our BBCI approach.

- Self-paced movements with the laterality controlled by the experiment conductor and without effect observation. This specification with most strong constraints should finally lay to rest all doubts of the most skeptic observer over the control of the application. The experiment conductor provides the user, in addition to the specifications previously

described, with the laterality of the next movement to be executed via an auditory command. However, the user decides by herself/himself when to act. Moreover, the user is neither able to observe the events of the feedback animation on the screen nor does she/he have any influence on the order of commands. The majority of trials here have also been successfully finalized with immediate as well as with the delayed reaction trials.

A series of single trials acquired throughout the entire experiment (here 64 left and 64 right trials) lend themselves to portray a summary plot (see Figure 34). The data is acquired in an experiment with self-paced and self-lateralized movements with observation of the feedback animation. Here, crosses were replaced for reasons of clarity by bold dots and the history tails are illustrated with bold lines for the four most recent periods and with thin lines for another four preceding periods (each period is lasting over 40 ms). The abscissa represents the classification results of the determination classifier, while the detection classifier's output value is indicated on the ordinate axis. It can be seen at first glance that the majority of trials were classified correctly.

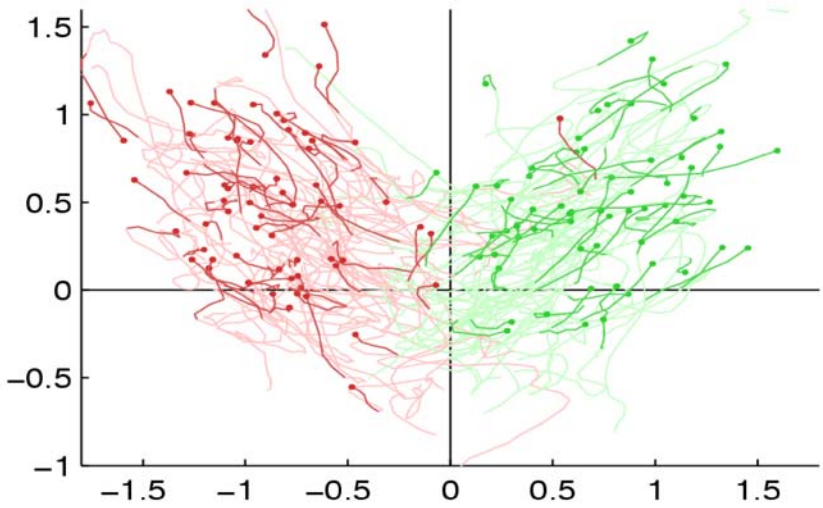


Figure 34: Accumulated trials from the "Jumping Cross" feedback scenario. Left trials (red) and right trials (green) are finalized with dots trailing history tails. The majority of single-trials are finalized in the correct quarter of the coordinate system, spanned by the output values of the detection (abscissa) and the determination (ordinate) classifiers.

Similar, so-called "*Spermiogram*" plots also result from the two remaining specifications of the self-paced experiment.

7.1.b Externally paced experiments

In contrast to experiments with intrinsically executed movements, experiments with imagined or phantom movements cannot produce any response event markers. This holds true for, at least, experiments with paralyzed patients. To overcome this problem, the user is provided with external pacing as well as lateralization from the BBCI system during special training sessions. In this way, detection classification becomes superfluous, since the pacing is predefined by the system. A purpose-built feedback application was implemented containing all

elements of the previously described feedback scenario for self-paced experiments, but; in this application the cursor movement is restricted in two ways:

- The cursor's vertical movement is fully pre-defined by a continuous slide from bottom up with a constant speed. This slide starts at the time stamp of the lateralization stimulus marker (indicating which action should have been performed next). However, this marker information is not displayed to the user in feedback-based application sessions; the user thus selects the laterality on her/his own accord. The cursor speed is calculated such that it finishes its movement at the upper margin of the screen at the pacing stimulus marker, provided by the digital metronome.
- The cursor's horizontal movement is bounded by a cone in such a way that its freedom is "scrunched" to zero at the beginning of the trial and increases continuously with its ordinate position until the full range of horizontal movement is offered at the end of each trial. The cursor's movement can be determined by multiplying the current determination classifier result with the radius of the cone, the latter of which depends on the cursor's ordinate position, i.e. the time elapsed since the first stimulus marker. The exact form of the cone can be defined by the implementation of the feedback module.

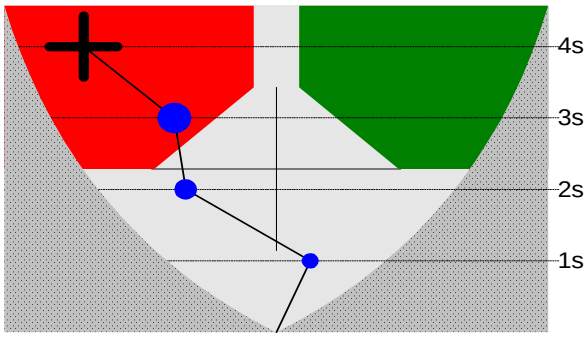


Figure 35: Schematic illustration of the feedback scenario for an externally paced experimental setup. The horizontal movement freedom of the cursor (i.e. cross) is restricted by the grey areas. The vertical movement is triggered by the event markers; it starts at the bottom with the stimulus and slides with a constant speed upwards. The cursor's deflection can be controlled by user's intentions

The feedback scenario for externally paced experiments is depicted in Figure 35 with four long-term history intervals of approximately 1 second each, corresponding to 25 command blocks of 40 ms each. Please note that this is not a screenshot, i.e. it is not presented to the user, and serves rather as a diagram that describes the behavior of the cursor. Therefore, neither the grey and dashed areas of the cone, nor the dashed time lines are visible to the user. They merely indicate the space where the cursor is not able to be located at certain time steps.

7.2 Feedback scenario "Brain-Pong"

Several gaming feedback applications were then investigated, three of which are described in the following subsections. The simple idea of one-dimensional control is to steer a racket from left to right in the lower part of the screen, with the aim of hitting a ball that bounces against the frame of the screen window. The first prototypes were named "Arkanoid",

dedicated to the well-known arcade computer game of the Atari and Commodore computer era. However, its name was then changed to the even older game of Tele-Tennis (also known as "Pong", Atari 1972). This ping-pong like brain game can be played by a single player, as shown in Figure 36, but also by two players competitively. The user is presented with a full-screen window of a neutral background color (e.g. light gray) that has a fixation cross in its center. A thick horizontal bar, taking up a fraction of the window width – the racket – of some contrasting color (e.g. dark gray, black or white) is placed on the bottom of the screen. The proportion of the width of the racket to the width of the window can be changed in the initialization file and defines the complexity of the game. The racket can be moved to the left or right by the user's intention. A yellow ball slides continuously from the current racket position upwards, and bounces off the sides and top of the window. The upper-right corner contains the game score showing hits in green and misses in red. The aim of the game is to move the racket so as to prevent the ball from dropping off the screen for as long as possible. If a ball is missed, i.e. it crosses the imaginary lower border, the simulation finishes and a new ball is initialized at a random position with a random but upwards velocity vector.

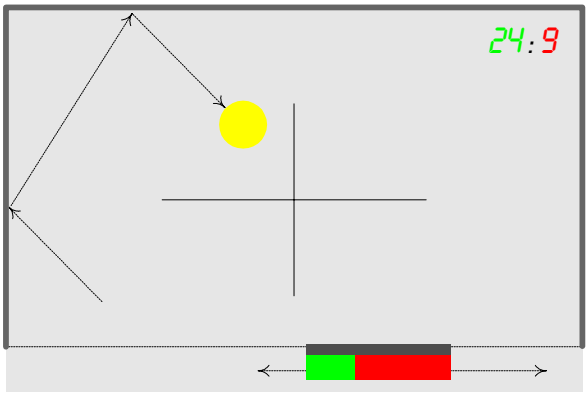


Figure 36: Schematic illustration of the feedback scenario "Brain-Pong". The ball (yellow circle) is sliding over the window being bounced off its borders. The racket can be controlled by the user's intentions to the left or right. The user achieves a bonus point for hitting the ball with the racket and is penalized if the ball crosses the dashed bottom line.

Figure 36 illustrates this feedback scenario; it should be noted that the ball's trajectory arrows, the lower margin of misses, and the racket's velocity vectors do not appear in a regular user display. The coloring of the racket can be realized in several ways:

- The three-color system is based on the three classes all samples are affiliated to during the classification process. In particular, this can be described by two lines: the filling line and the discrimination line. The horizontal filling line is drawn at a certain height of the racket. That height is negatively correlated to the fuzzy value of the affiliation certainty of the currently processed sample to the rest class. Such that, the line will be positioned higher, the smaller the rest fuzzy value is. The area under the filling line is divided by the vertical discrimination line, which is drawn at the position given by the difference between the fuzzy values of the class of right movements and left movements. It will therefore be positioned further to the left, the higher the fuzzy value of the left class is for the current interrogated sample and vice versa. Please note that the crossing point of these lines is equivalent, up to a scaling factor, to the coordinate of the cursor in the "Jumping Cross" feedback scenario, and that only the colors of its target fields are exchanged in

their positions. However, a larger region will be colored red as the line moves to the left, indicating a left-movement. The opposite holds true for the green-colored area.

- The use of color filling for the preferred class is designed similarly to the above-mentioned filling strategy, although it lacks the discrimination line. The area under the fill line is colored either red or green, depending on which fuzzy value is greater. As a result, the user is able to recognize the most probable action class, i.e. the one with the larger fuzzy value, from the color of the racket.

- The final coloring strategy employs a smooth change in the filling color. This additionally lacks the fill line, thus the complete racket area is filled with a single color, continuously changing its RGB value, for example from red to green. However, this strategy may result in intermediate colors that the user is not able to associate with any of the classes of interest, which can then be a source of confusion.

These filling strategies can be incorporated in the feedback animations for users unfamiliar with this kind of brain-gaming. The intent here is to provide these users with information concerning their intentions: by learning to monitor the coloring, they will be able to select an appropriate linkage strategy and gain insight into the signals' behavior. In further experiments, when the user had seized sufficient control over the feedback application, the filling strategy proved to be rather disturbing. For this reason, coloring strategies were not employed in advanced "Brain Gamers".

Several subjects who used the BBCI system to play "Brain-Pong" reported several long phases with many successive successful trials. This provided them with the feeling that the racket was becoming integrated as a part of their own body and that no particular effort was required to maintain control. Moreover, it was reported that in successful phases, performance improved even more, whereas in careless phases filled with mismatching trials, performance dropped drastically. These observations are also evident from the neurophysiological and machine-learning point of view. Since the user is not put under pressure during training sessions, she/he generates ordinary brain patterns that are fed into the learning machine. However, during application sessions with feedback, the user may become greedy and at some point generate different brain patterns induced by an additional effort to re-seize control over the feedback. To conclude, the user puts herself/himself in a conflicting situation by applying more effort to gain control; the effort induces an immediate change in the brain pattern feature space, spanned by the training data and thus recognizable by the classifier, which results in losing even more control. "Brain Gamers" further report that when control performance worsens significantly (subjectively measured), simply adopting a relaxed state or indifferent attitude to the game helps to re-gain control over the game.

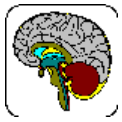
Two different control strategies were implemented and tested with this kind of feedback. While both are appropriate for this scenario, other control strategies may be preferable for other applications. The following two subsections describe both these control strategies in further detail.

7.2.a Stepwise displacement control strategy

In the *stepwise* or *staged displacement* control strategy, the racket is moved by a fixed displacement step. The step length is defined by the percentage of the window or racket width the racket is repositioned in the direction indicated by the class label, e.g. left or right, when a particular command is recognized as being affiliated to the one or the other class of interest for at least the CAT duration. The racket retains its position if a rest-command is detected for at least the CAT duration, although in this case the system does not fall into the relax mode. The racket also remains stationary when no stable signal is present, i.e. when the combined classifier output indicates more than one class label within a single CAT duration. This control

strategy is based on the CAT and CRT parameters and is described in part at the beginning of this section on page 51 ff.

Since the command emission system falls into the relax mode for the CRT duration after each emission of a command, the user might expect difficulties repositioning the racket fast enough from one side of the screen to the other. The racket could thus “receive” a reposition command with a minimal interstep interval given by CRT-CAT+1. Consequently, a CRT set to a value only slightly greater than or even equal to CAT will solve this problem. It will therefore also generate a greater amount of false-positively emitted commands, making it rather difficult to reposition the racket by only a single step to the left or right, since this setting prefers longer trains of similar commands.

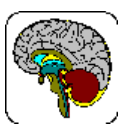


Notably, the Lateralized Readiness Potential (LRP) can indicate preparation for one particular movement. A series of movements and movement repetitions in particular may not necessarily be linked with a LRP of the same strength for every single movement. Rather, the LRP might index primarily the start of the whole series, invoking subcortico-cortical routines for the execution of the repeated actions. This insight points to a potential limitation of the *stepwise displacement* control strategy: as in the task of quickly relocating the racket, the user will need to emit a series of identical commands. Identifying the optimal EEG correlates for such a hyper command will require further studies.

7.2.b Graded deflection control strategy

Another control strategy utilizes, in addition to the class label of the sample (that of the most probably assigned class), the fuzzy value of the most probably assigned class in and of itself. The latter exploits the strength of the recognized command signal (e.g. a measure of the LRP amplitude) and thus allows a *graded deflection*. From its initial location at the center of the screen, the racket can be deflected laterally outwards. However, as it is attached to an imaginary spring, it will be returned to its initial point (i.e. inwards) whenever the fuzzy values of neither the left nor right classes are significantly high. Accordingly, this control strategy employs only a virtual rest class, whose fuzzy value can be calculated by combining the results of the left and right class fuzzy values. Thus, the fuzzy value rises following a decrease of both the remaining class fuzzy values. It should be noted that this is consistently regarded the terms of fuzzy logic, since the fuzzy values are calculated from the outputs of independent classifiers and thus do not necessarily sum up to one.

The outward racket deflections are calculated each time a new result data block is received from the classifiers, i.e. at a frequency of 25 Hz. Each deflection is proportional to the difference between the momentary graded values of the two action classes, but only where the graded value of the virtual rest class is below a certain, usually low, threshold. Otherwise, it is set to zero, such that the racket is almost always on the move. The value of the rest class threshold controls the trigger sensitivity for the displacement activation and can be adjusted together with the proportionality constants for the racket’s outward deflection, depending on the user’s experience and preference. Notably, the proportionality constants for the racket deflection to the left and right should be adjustable separately. This is reasonable because it is simpler to generate movements of the dominant hand; they in turn demand lower amplitude of the LRP on the contralateral hemisphere of the brain. Since the classifier tries to balance out this discrepancy, subjects often report arbitrary and indeterminate periods of right or left overweighting of the output, resulting in the racket seeming to be glued to one edge of the screen.



A large variety of other neurophysiological paradigms besides LRP can be employed to control feedback applications that rely on this control strategy in a similar manner. Among these are Event Related (De-)Synchronization (ERD/ERS) and numerous variants of the brain state paradigms. Most paradigms are capable of extracting not only a discrete value of the class label, but also the signal strength as a real numeric value. Incorporating this measure in feed-

back applications in a direct way, i.e. not post-processed by command emission modules, can improve the BCI user's capability to gain more accurate control over the animation object.

7.3 Feedback scenario "Brain Pacman"

After successive experiments with the "Brain-Pong" game as the BBCI feedback application, many questions arose concerning user behavior with other kinds of feedback applications. The need to further explore various brain-gaming scenarios was thus recognized, which led to new experiments with further brain-gaming feedback applications.

In this the next step, the well-known Pacman video game was adapted to serve as the feedback application. The idea here is to creatively combine information from the "Jumping Cross" feedback application with a command-based control strategy in a purposeful gaming application. A random labyrinth was generated on a full-screen window with a neutral background color. The rules for maze generation are summarized in the following steps and exemplified in Figure 37:

- Step 1: The space for the labyrinth must be a matrix with odd dimensions. Index lines and columns of the maze start with 0. Figure 37 exemplifies the generation of a maze on a matrix of 11×7 cells.
- Step 2: Surround the space of the maze by wall cells, i.e. draw two horizontal walls at Y-positions 0 and 6, and draw two vertical walls at X positions 0 and 10.
- Step 3: Place the entrance door at a random but even Y position and at X position 0; place the exit door at a random but even Y position at X position 10.
- Step 4: Create a sub-labyrinth between the most upper-left and most lower-right positions, (0; 0) and (10; 6), by performing the following sub-steps where possible. If the width or the height of the space for the sub-labyrinth are less than 3, indicate that the creation is finished (exit the recursion and continue at Step 5).
- Sub-step 4a: Calculate the proportion `pr` of the width and the height of the sub-labyrinth and choose randomly between vertical or horizontal separation, weighted however by the proportion `pr`, such that horizontal separation is preferred for `pr<1` and vice versa.
- Sub-step 4b: Draw the selected type of the wall at a random but odd position through the complete sub-labyrinth, which will divide this into two sub-labyrinths.
- Sub-step 4c: Place a door within the wall drawn in sub-step 4b at a random but even X or Y position, for horizontal or vertical separating walls respectively.
- Sub-step 4d and 4e: Consider the two newly created sub-sub-labyrinths emerged from the separation of the sub-labyrinth recursively, as described in Step 4, i.e. in two recursive calls.
- Step 5: Find the exactly one existing shortest path from the entrance to the exit by the back-tracking strategy and mark this with step marks.
- Step 6: Place a certain number of bonus marks, e.g. apples, at random but "free" cells of the labyrinth. The number of bonus marks can be specified in the initialization file. Finally, place Pacman's initial position at the entrance of the maze.

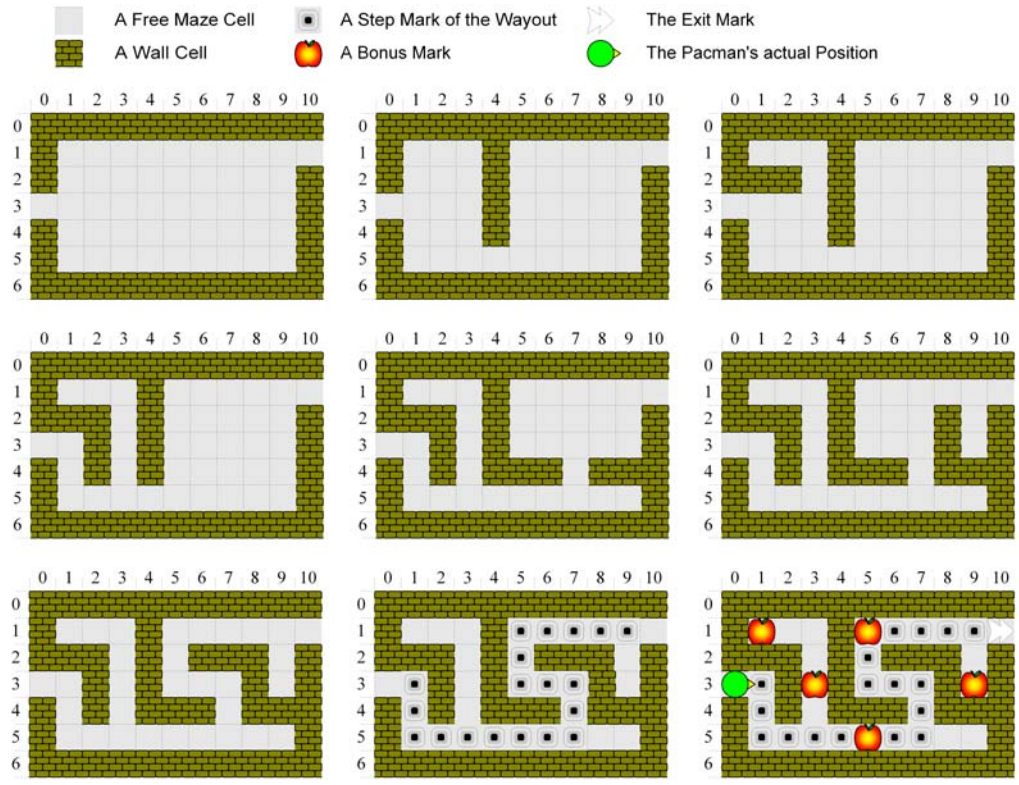


Figure 37: Steps in labyrinth generation process for the “Brain-Pacman” feedback scenario as described above. A binary tree-recursive algorithm is employed that generates a maze with exactly one shortest path from the enty (in the left wall) to the exit (in the right wall). The path is indicated by stepmarks. Apples picture additional bonus points.

The player can navigate Pacman any way through the labyrinth. Thus, although the aim of the game is to navigate Pacman through the maze from entrance to exit, the player can deviate off the path indicated by step marks in order to harvest apples and receive an additional bonus. Step marks proved helpful in larger labyrinths. The control strategy here employs the following approach: Pacman makes one step with each inter-step interval (defined in the initialization file, e.g. every 2 seconds) in the direction its nose is pointing until it faces a wall. There, it remains still, waiting for control commands to make a right or left turn.

Initially Pacman’s head is white and becomes filled with red or green color from the bottom up as the player’s intention to turn rises, i.e. the detection classifier yields action results. A vertical line discriminates the two colors and is placed according to the determination classifier result. However, other filling strategies can be considered as equally effective, e.g. those described for the “Brain-Pong” game on page 71. If the player intends to steer Pacman to the right, this should fill Packman’s head green for at least the CAT duration, and vice versa. After a turn, Pacman does not accept any further commands for the CRT duration.

Walking through the maze, the player acquires one bonus point for each move on the step mark and loses one point for each unsuccessful move against the wall. Furthermore, she/he can acquire bonus points for each apple. The simulation is finished when Pacman reaches the exit of the labyrinth.

A healthy subject will be able to navigate Pacman through the labyrinth presented above within 40 seconds (20 steps, each two seconds long) using a conventional keyboard or mouse. Using brain control only takes considerably longer, although the “fun-factor” of navigating

Pacman solely by the intentions of one's brain has turned out to be very appealing. Moreover, of great interest is that when immersed into the Brain-Game scenario, users sometimes had the feeling that Pacman turned in the correct direction although they were not consciously aware of having made that decision, sometimes not even ready to make a decision. This observation can be explained by the fact that the user generates brain patterns in advance, i.e. before Pacman reaches a desired turning point. The corresponding control commands are thus emitted before the user anticipates her/his intrinsic command emission.

7.3.a Turning if reasonable

A more intelligent control strategy was developed in the later version of the "Brain-Pacman" feedback scenario. It incorporates early command emission and stores the last turn command within Pacman; the pointing direction of Pacman's nose determines the direction of the next turn. However, Pacman continues moving in the current direction until the stored turn is sensible, at which point Pacman actually performs the turn. At that point, the most recently stored command is deleted by the recognition of the opposite one and Pacman is turned by 180° when the same command is repeated.

That control strategy incorporates pre-knowledge about the environment surrounding Pacman, i.e. the maze, and provides the user with a control mechanism that more resembles her/his natural behavior in certain environments. Consequently, this kind of assistant control system can be integrated into a wheelchair, allowing the handicapped person to indicate his/her next intended turn by an ongoing or blinking yellow turn signal similar to those used by conventional vehicles. The wheelchair should therefore be furnished with cameras that examine the surrounding environment and be connected to a system able to detect turning possibilities. [Millán and Mouriño, 2003]

7.3.b Feedback scenario "Brain-Turnman" – a Pacman-variation

The last variation of the "Brain-Pacman" scenario was developed especially for one-class experiments, where the subject is able to initiate only a single kind of action, in contrast to mere resting or relaxing. This experiment should demonstrate that it is possible to gain control over a complex feedback application with a minimum of recognizable classes, although the control speed is compromised as a result.

The user is presented with the above-mentioned graphical setup that differs only in that Pacman does not make any move without the recognized control command. Therefore, Pacman's head turns continuously clockwise by 90° with each inter-step interval, e.g. every two seconds. The emitted control command prompts Pacman to make a step in the direction of its nose, followed by a reset of the inter-step interval counter, such that a rapid repetition of a single command would result in a fast run along a corridor. When turning Pacman, the player should pause at the intersection and wait until its nose points in the direction of the next move.

Besides the requirement of a small CRT, this technique is expected to be more stable and thus helpful for users whose multi-class paradigms lack satisfying classification results. Classification results are commonly known to worsen with the increase of classes of interest; this may stem from the poor spatial resolution of EEG-based acquisition techniques.

In conclusion it is recognized that the "Brain-Pacman" feedback application still shows difficulties in controlling an object in its own coordinate system. Moreover, the already mentioned need to explain brain-gaming scenarios in the user's coordinates system, not in that of the actor, was confirmed. This would allow the user to get inside the animation, where she/he could gain control by a "natural mapping" of the action required in the virtual reality or gaming scenario to the "action space" of the human operator, which is coded in egocentric coordinates.

Thus, a Pacman scenario where the user navigates through a labyrinth, but is provided with an animation from Pacman's point of view, i.e. inside the labyrinth, might be more appropriate and yield more satisfying results. However, this is an issue requiring expert knowledge in the field of Virtual Reality (VR).

7.4 Feedback scenario "Brain-Tetris"

Feedback scenarios suited for multi-class paradigms represent a final series in the collection of feedback modules. Control over this gaming application is very complex, yet its control strategy is very simple, similar to those of other command-based feedback control strategies. Tetris is a well-known video game invented by Russian programmers in the mid-eighties. It consists of bricks assembled each of four pieces that fall from the top of the screen at a certain speed, i.e. the inter-step interval. The aim of the game is to place these bricks, so as to complete the lines at the bottom of the playing field; the completed lines then disappear. While a brick is falling, the animation module waits for a control command with one of the following class labels: (*i)* move brick one step to the left, (*ii)* move brick one step to the right, (*iii)* rotate brick by 90° clockwise, and (*iv)* drop brick down as far as possible. The player earns different credits for every brick put in place and for the number of lines completed. Once a single horizontal line or multiple lines are complete with bricks, they are deleted and all bricks above them shift down. The aim of the game is to continue for as long as possible; the game finishes when it becomes impossible to initiate a new brick on the top of the playing field without it covering a previously placed brick. Please refer to Figure 38 for a screenshot of the feedback scenario "Brain-Tetris".

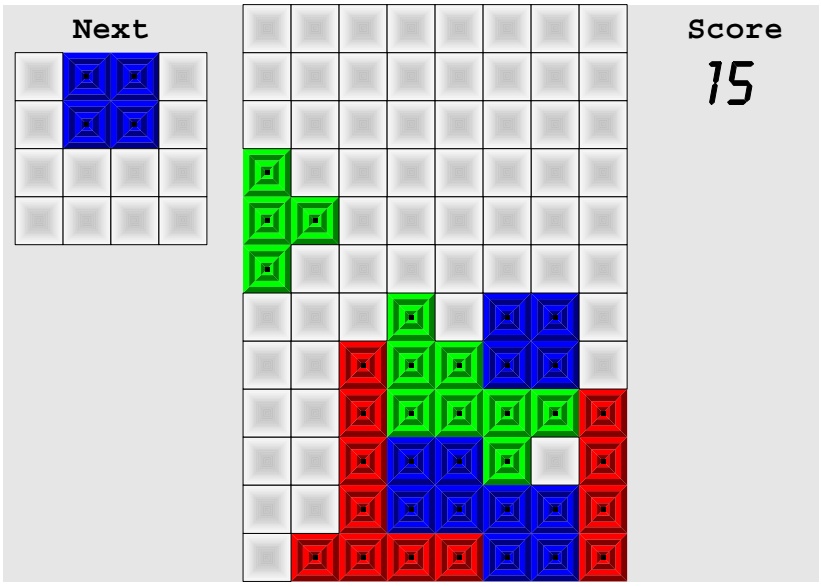


Figure 38: Screenshot of the feedback scenario "Brain-Tetris". Bricks that consist of four pieces are falling down the play field stepwise. The user can control the brick's movement stepwise to the left or right, its rotation (90° clockwise) or she/he can drop the brick to the lowest possible placement. The next brick and the user's current score are displayed.

For purposes of control in this application, four different brain states were adopted, each assigned to its own class of interest. With this feature, it becomes second nature to encode the left movement of a brick by an imagined movement of the left hand and, accordingly, the right movement of the brick by an imagined movement of the right hand. Furthermore, an imagined foot movement was employed to encode the drop function, since the player's "down" intention is naturally somewhere in the lower part of the body. To encode the rotation of the brick, it seemed most appropriate to have the user imagine a mental rotation of some non-trivial 3D-figure, since the correlate of visual intention is provided by the user's desire to see the Tetris brick from some other angle. However, this approach was restricted to a simple 2D rotation of a Tetris brick. The expectation that this would invoke a strong desynchronization in firing rates of large ensembles of cortical neurons over the visual cortex (located in the parietal to occipital regions of the brain) was proven true. Since the visual cortex is further back in comparison to the sensorimotor cortices of hands and feet (located over the central sulcus), it can be satisfactorily discriminated from the other classes.

To allow for more comfort during the experiments, i.e. to reduce the user's waiting times between sessions and to simplify the experiment conductor's effort to reset the feedback parameters, a remote control unit was employed so that all parameters of the game simulation could be reset during the online operation of the game and from a different computer. With one click, the experiment conductor can send the parameter package to the application, where it takes effect as soon as it becomes possible for the application to commit the changes. The entire game scheme can be changed in this remote control application by manipulating parameters such as:

- Command emission parameters CAT, CRT and the Command Activation Threshold (CATh). In addition, the "Brain-Tetris" sends out its states to the parallel port of the feedback computer, which can be connected to the recorder PC and store arrived data as corresponding event markers. This permits a later analysis of data acquired during the feedback sessions, through which entire games may be reconstructed.

- Game behavior: The countdown begins when the game starts, i.e. before the first brick starts to fall, providing the user with a relaxation period that sometimes helps her/him prepare for the simulation. The inter-brick interval lets the user revise the completion of a brick before the next brick is presented. Different timing and design options, such as game field dimensions and bitmaps, are used to construct each brick.

- The capability to switch off certain types of bricks. This, in a sense, controls the difficulty level of the game. Each brick type can be assigned a separate score; the user then gains that score for positioning the brick, in addition to the score for completing a row.

- Class labels: This adapts the game to the labels sent out by the BBCI system. The experiment conductor then no longer needs to adapt the complex BBCI applier, but can easily reassign the class labels or simply switch off the recognition of a certain class.

- Finally, all these parameters can be saved to an initialization file and loaded in later experiments.

Unfortunately, however, this particular game appears too complex to be controlled by current BCI systems. In certain game situations (e.g. when the user positioned and rotated a brick as if planning to position it on top of other bricks), a drop pattern of the brain is erroneously detected as a rotate pattern, resulting in an error with disastrous consequences for the entire game. Taking, for example, the game situation shown in Figure 38, a correct recognition of the drop pattern would result in a completion of two rows, while a misclassification of the brain state as a rotation command would impede the completion of the five lower rows assembled with great care by blocking them from the top. In summary, the rules of the "Brain-Tetris" feedback application are not well suited for brain-gaming, although in one experiment a user

was able to complete eight rows before becoming stuck. Nevertheless, “Brain-Tetris” continues to attract attention from many who are fascinated by the notion of the application.

7.5 Feedback scenario “Mental Typewriter”

The last two feedback applications presented in this work are rehabilitative, i.e. employed in experiments that are designed especially for handicapped users. These applications allow these users to communicate with the surrounding environment and to control computer applications. To keep matters as general as possible, a spelling application was developed that can be controlled through a command-based strategy.

The user is provided with a full-screen window that is subdivided into three parts: One is the full screen width field in the upper half of the window, which contains the spelled message from the user. The other two parts are selection fields in the lower part of the window, each containing their own parts of the alphabet and control symbols that can be selected by the user during the spelling process. Of particular importance is that the alphabet, together with the control symbols, was designed as a binary tree to be integrated in the application. Moreover, to allow for a fast selection of a letter or symbol, the letters located on the leaves of the tree are weighted by their probabilities of occurrence in a certain natural language. In this way, the heavier a leaf in comparison to all other leaves, the shorter is the path from the root of the tree to that leaf.

The probabilities of letter occurrences were calculated from the analysis of texts that were randomly selected from the Internet. Texts in German, English, French, Spanish, Italian and Russian were analyzed and results are presented in Table 7. Similar results for letter occurrences can be also found in the literature on military cryptography and cryptanalysis since Turing’s time, e.g. [Friedman and Callimahos, 1920] and [Черёмушкин et al., 2001] for Russian. The procedure for generation of a binary tree from a table of probabilities is mainly guided by the constraint of the optimality mentioned above. However, it must yield a tree where the letters retain their position defined by the order of the alphabet, such that a novice user is not confused by an obscure order. This constraint will enforce the procedure to yield a sub-optimal tree solution only. It thus proved to be more natural for the user to navigate a binary tree in an already familiar order, than to chase for information of the order of small bit fractions in a coding that is rather difficult for a human operator to retrace.

Table 7: Probabilities of letter occurrences in texts of different languages

Latin Letter	Occurrence in the corresponding language [%]						Cyrillic Letter
	German	English	French	Spanish	Italian	Russian	
A	5.52	7.96	7.68	12.90	11.12	0.062	А
B	1.56	1.60	0.80	1.03	1.07	0.014	Б
C	2.94	2.84	3.32	4.42	4.11	0.038	В
D	4.91	4.01	3.60	4.67	3.54	0.013	Г
E	19.18	12.86	17.76	14.15	11.63	0.025	Д
F	1.96	2.62	1.06	0.70	1.15	0.072	Е, Ё
G	3.60	1.99	1.10	1.00	1.73	0.007	Ж
H	5.02	5.39	0.64	0.91	0.83	0.016	З
I	8.21	7.77	7.23	7.01	12.04	0.072	И, Й
J	0.16	0.16	0.19	0.24	< 0.01	0.028	К
K	1.33	0.41	< 0.01	< 0.01	< 0.01	0.035	Л
L	3.48	3.51	5.89	5.52	5.95	0.026	М
M	1.69	2.43	2.72	2.55	2.65	0.053	Н
N	10.20	7.51	7.61	6.20	7.68	0.09	О
O	2.14	6.62	5.34	8.84	8.92	0.023	П
P	0.54	1.81	3.24	3.26	2.66	0.04	Р
Q	0.01	0.17	1.34	1.55	0.48	0.045	С
R	7.01	6.83	6.81	6.95	6.56	0.053	Т
S	7.07	6.62	8.23	7.64	4.81	0.021	У
T	5.86	9.72	7.30	4.36	7.07	0.002	Ф
U	4.22	2.48	6.05	4.00	3.09	0.009	Х
V	0.84	1.15	1.27	0.67	1.67	0.004	Ц
W	1.38	1.80	< 0.01	< 0.01	< 0.01	0.012	Ч
X	< 0.01	0.17	0.54	0.07	< 0.01	0.009	Ш, Щ
Y	< 0.01	1.52	0.21	1.05	< 0.01	0.014	Ъ, Ь
Z	1.17	0.05	0.07	0.31	1.24	0.016	Ы
						0.003	Э
						0.006	Ю
						0.018	Я

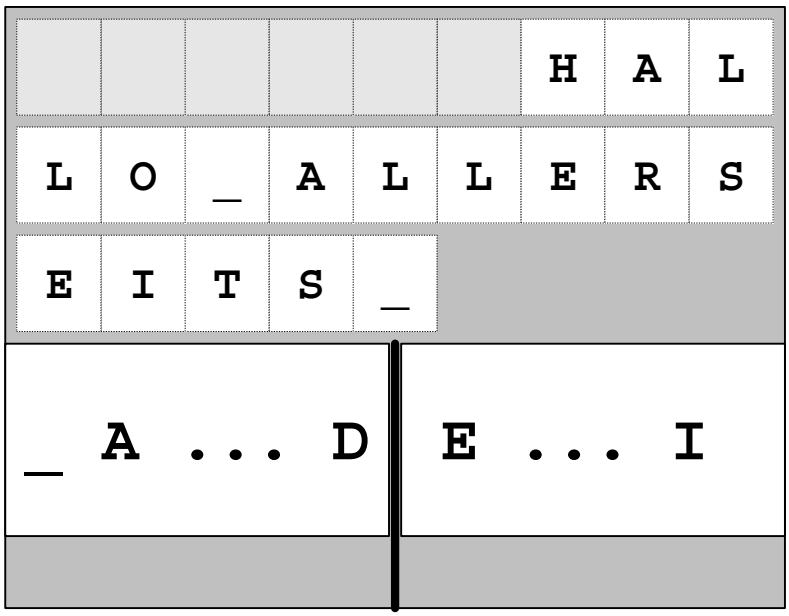


Figure 39: Screenshot of the feedback scenario “Brain-Speller” in an intermediate selection processing stage. The two selection fields contain parts of the alphabet. The user selects either left or right one that contain the desired letter by imagining or executing the left or right hand movement. This allows “traveling” across the binary alphabet tree until a letter is selected; it is then appended to the typed text.

The software engineering approach for tree generation is described in more detail in section 3.3.d of the next chapter. Figure 39 shows a screenshot of the feedback scenario “Brain-Speller” at an intermediate stage. The user is initially presented with an empty message field and the alphabet. This alphabet is subdivided into two halves according to the structure of the generated binary tree, i.e. based on occurrence probabilities; these halves appear respectively in the left and the right selection fields. An animation control command emitted by the *Queue Checker* Module prompts an action either to select the left or right selection field and to perform a single step down in the binary alphabet tree. The user screen then changes according to the new selection processing stage that is defined by the position in the tree.

- If the user selects a field containing more than one letter or symbol, it is enlarged in a fast animation to fit both selection fields. Both intersection letters are then indicated, such that the range of letters is divided once more on the both selection fields. After the CRT period, concluding the selection trial, the user is able to perform further steps until only one letter or symbol is contained in one or both selection fields.

- If the user selects a field containing only one letter or symbol, i.e. the corresponding leaf of the alphabet tree, the corresponding letter is drawn in a fast animation upwards to the end of the message field, which shifts all preceding letters one position to the left.

Selection of the “space” character (‘_’) inserts an underscore symbol into the message. Nevertheless, selecting the backspace character (‘<’) erases the most recently selected character from the message, thus shifting it one position to the right; note that the placeholder for the next selected symbol is always in the center of the screen, as it is common for the mechanical typewriter.

Several variant selection strategies are conceivable for this operation and could be implemented in a simple manner. For example, the user may feel more comfortable with being prompted to confirm the selected the letter, such as “Insert Letter ‘X’ into the Message?” This can be displayed in an additional field below the selection fields together with the indication of “Yes – Correct” and “No – Error” on the selection fields. In all cases, regardless of the update of the message string, the user is presented with the initial screen, again with the complete alphabet, and can start selecting the next letter or symbol from the beginning where the selection stage is positioned at the root of the tree.

Calculating binary alphabet trees is computationally inexpensive and can thus be performed with every update of the message string, such that an adaptation of the tree to the user’s spelling preferences is an optionally built-in function of the speller application. A disadvantage of the automatic adaptation is that during long sessions the user may become familiar with certain selection sequences of the most frequently used letters, which will then change until the system settles. Of interest in further studies is the analysis of letter co-occurrences, which is obviously also language-specific. The system can provide the user with the most frequent letters in a modified binary tree, which is subject to the letter previously selected by the user. In some cases a word completion function may be useful for the real-world application of the system. This function can also keep record of the occurrences and co-occurrences of letters preferred by the user. In addition it can store, as a dictionary, the most frequently spelled words, and propose these to the user for completion.

7.5.a Built-in error detection

Since the emitted control command may in some cases be erroneous, there is an evident need to equip an application such as “Brain-Speller”, where it is difficult to make a back step, with an optional error detection and correction paradigm. Section 6 described error signal detection directly from the user’s brain signals in terms of the benefit that can be additionally gained from this paradigm. The “Brain-Speller” feedback scenario, employing a binary alphabet tree, allows for even more possibilities than simply canceling the most recently emitted command, in the sense that it can then correct the user’s operation in an intelligent manner. This saves time, since the user does not need to repeat the erroneously recognized operation.

When the classifier erroneously detects and emits a certain command, the feedback module processes it and thus confronts the user with an unanticipated state. The user is then not able to identify herself/himself with the present situation and assumes the recognition module having produced an erroneous decision, i.e. the user becomes displeased and annoyed with the most recently performed action. This reaction is monitored after every emitted command for a certain time period, e.g. 500 ms, such that the brain signals are checked for error potentials as described in section 6. If no error potential is recognized within a given time, the action is confirmed and no change in the feedback animation is required. The confirmation is then communicated to the recognition module, which then switches to recognizing further control commands. In the opposite scenario, i.e. when the user’s reaction contains an error potential, the feedback animation may react in one of two ways: (*i)* by canceling the most recently performed operation and returning to the previous state, known as good. In the current “Brain-Speller” application, this is one step upwards in the binary tree. (*ii)* The last operation can be reversed, although that option is only conceivable in binary decision applications such as the “Brain-Speller” feedback scenario with only two selection fields. In order to decide what the appropriate feedback reaction should be, the data forming the basis of the action decision (of a particular past period) should be reanalyzed by applying a special treatment.

As mentioned above, one feedback module’s decision about the emission of a command is integrated throughout the CAT duration of classifier’s decisions, each of which can be categorized as a detection and a determination classifier decision. To emit a control command to the animation, the detection classifier must indicate a positive result, while the determination

classifier must provide a result that differs from zero significantly; both indications must last for at least the CAT duration. If the user, in turn, indicates that the last decision was erroneous, the detection classifier's results are reanalyzed under certain, somewhat more difficult, conditions to clarify whether a command should have been emitted. These conditions, for example, could distinguish themselves by greater thresholds and/or longer CAT periods. If the data fails the reanalysis, the most recently emitted command is simply canceled, and the feedback animation is redrawn according to the current, i.e. re-updated, state of the system. However, the reanalysis results of the detection classification may substantiate the correct emission of the control command; in this case the most recently performed action should be reversed, followed again by a redraw procedure of the feedback animation.

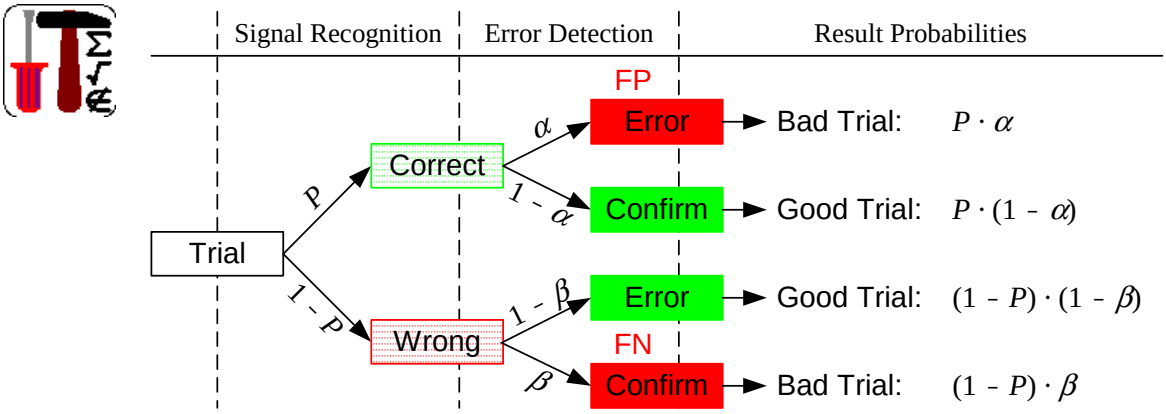


Figure 40: Signal and error detection procedure for a BCI system alone performing with the accuracy P, and the error detection module performing with α false positives and β false negatives.

Besides the probability P of the main data recognition module making correct decisions – aimed to be as high as possible, though without ever reaching the ideal `1.0` value – the error detection module is further contaminated by two kinds of errors: *(i)* the false-positive rate α, and *(ii)* the false negative rate β, which can be viewed in the same context as probabilities of incorrectness of error detection. The procedure of signal and error detection, together with the resulting probabilities following all good and bad classified trials, is illustrated in Figure 40.

Consequently, the probability of a good trial after error detection can be expressed as:

$$P \cdot (1-\alpha) + (1-P) \cdot (1-\beta) = 1 + P \cdot (\beta - \alpha) - \beta \tag{19}$$

Thus, the error detection mechanism increases the performance of the BCI system if α and β are selected such that the inequality (20) holds:

$$P < \frac{\beta - 1}{\beta - \alpha - 1}. \tag{20}$$

It is obviously possible to derive the corresponding inequality for the case of endless error detection, i.e. if the user reaction is repeatedly monitored after correcting the feedback animation. However, as this is of theoretical value only, it is omitted from further discussion here.

7.6 Feedback scenario “Virtual Arm”

The final application to be presented in this thesis, the “Virtual Arm” feedback scenario, represents a feedback application par excellence. It could be employed within the scope of future research and experiments for use with patients in their final BBCI qualification stage, i.e. after they have learned about their own intentions, emission of control commands and have developed a certain own control strategy for high level operations, as well as know how to deal with the feedback animation. The “Virtual Arm” feedback scenario is based on Virtual Reality (VR), since the user is instructed to control a part of the body that may be absent and that is displayed on a computer screen. Readers with a scientific or technical background will know that rehabilitation engineering represents a vast scientific field that overlaps with medicine, prosthesis engineering, human psychology and many other important disciplines. It is to be noted that none of the assumptions made in this thesis should be taken as axioms. To minimize possible misreadings, I will limit descriptions to already completed research and to experiments concerning bio-feedback applications carried out within the scope of the BBCI project. The “Virtual Arm” feedback application described here still remains an open research field that requires prior solving of many neurological, technical and fundamental research problems, as well as design and conduction of further exhaustive experiments with patients.

During the training session the user is instructed to perform imagined movements of the arm, e.g. bending it at the proximal (shoulder or elbow) and distal (hand or fingers) joints, which are triggered by an external query. After the extraction of training samples, the data is preprocessed and fed into the learning machine. During the application session the user is presented with a black full-screen window containing an image of a human arm; the experimental setup defines the laterality of the extremity, i.e. left or right arm. The user’s monitor is positioned such as to maximize the impression in the user that the arm is her/his own, as indicated in the two photos taken from one of the experiments, in Figure 41.

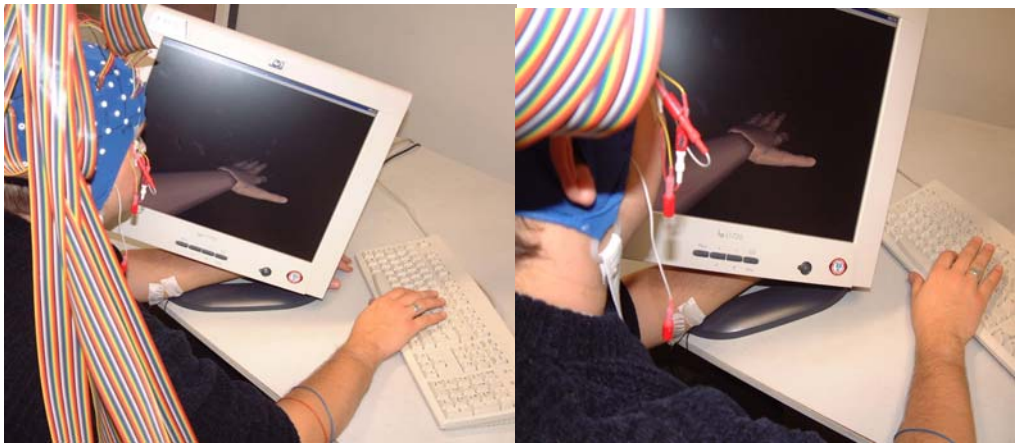


Figure 41: User and monitor position in experiments with the “Virtual Arm” feedback scenario. Monitor is placed and rotated such that the user’s imagination of observing the own arm while looking on the screen is maximized.

Here, the user was instructed to press a key on a computer keyboard with her/his right hand and to press on the desktop with her/his left hand according to auditory queries. The data was acquired in the following manner. When the digital metronome inserts a stimulus marker in the data, the keyboard correspondingly inserts a response marker. A sample extracted from

the continuous data, preceding a single stimulus marker, is thus denoted as a training sample for the phantom movement of the left hand, while the sample preceding the combination of the stimulus and the response markers, located in the data within a certain, short duration (50-100 ms), is denoted as the real executed movement of the right hand. The most accurate positions among these extracted samples of phantom movements can be ascertained from the shifts between the stimulus and the corresponding response marker, since once the user has found a right tact she/he tends to retain it throughout the entire experiment. Consequently, the phases should be approximately the same for all trials, irrespective of their lateralization.

The VR platform used to handle and manage the 3D objects of the virtual arm was kindly provided by the Visualization Systems, Technology and Applications (ViSTA) group at our institute and is based on a Demonstrator Software for the project X-Rooms [`http://www.xrooms.de`]. The control protocol for the arm movements was defined and kindly provided by Dipl.-Inform. Karsten Isacovic, director of the X-Rooms project. However, the control protocol proved incompatible with the communication protocol of the feedback interface of the BBCI system; an additional interface application was thus implemented to solve this problem.

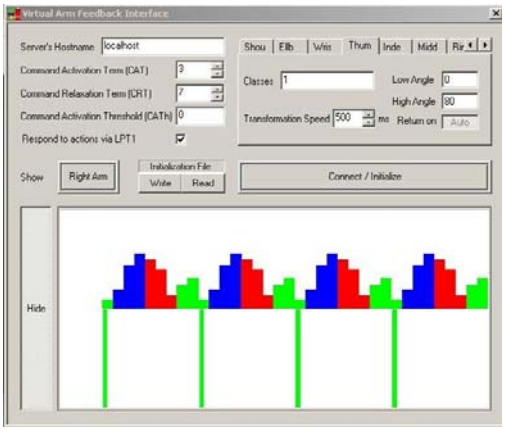


Figure 42: Screenshot of the "Virtual Arm" interface GUI. A variety of parameters can be set up in the upper part of the window, class labels and transformations can be defined for each joint separately. The canvas in the lower part of the window indicates control commands received from the BBCI system as colored bars and transformation commands emitted to the VR environment as thick colored lines; the colors denote class labels.

This interface needs to be as flexible as possible to ensure greater suitability for a variety of future, yet to be defined, experimental setups and allow as much "degrees of freedom" (DOF) as possible for arm bending and moving. The human arm consists of eight parts: shoulder, elbow, wrist and five fingers, each capable of two to three DOF. This means that the entire arm is capable of about 22 mostly independent DOF. The arm model in the VR environment is capable of even more DOF, although some of these are impossible in terms of human physiology; it was therefore necessary to reduce the number of DOF to a minimum that is based on our neurophysiological approach. It is appropriate to allow no more than one DOF for each joint so that the interface application can distinguish between the three main joints of the arm and the five fingers, assigning to each of them a class label. The start and stop angle together with the transformation time allow to create an animation of the corresponding joint of the virtual arm bending in a VR environment. The *return type* specifies whether the reverse animation is initiated: automatically upon completion, upon the emission of the rest command, upon

repeated emission of the same control command, or if at all. Besides the usual control command emission parameters such as CAT, CRT and CATh, it is notable that the "Virtual Arm" interface application can run on separate computer. It does not have to run on the VR-application or BBCI applier system and communicates via a Local Area Network (LAN) that relies on the UDP/IP protocol. Figure 42 illustrates the screenshot of the Graphical User Interface (GUI) of the "Virtual-Arm" interface application. Please note the additional summary of the incoming command blocks (colored according to their class labels) on the upper half of the canvas, and the emitted animation commands (with bold lines in the color of the to-be-updated current joint) in the lower part. This graphical display can be switched off to save computational resources and processing time, for example, when the interface is running on the same computer as the BBCI applier.

7.6.a Controlling grasping and arm bending

Let us assume that a BBCI user is capable of producing two different spatio-temporal brain patterns that concern distal and proximal joint flexions of a limb. At the current stage of research, it is only of marginal interest whether the user is intrinsically capable of controlling this limb or not, i.e. the limb could be actually paralyzed by a neurological disease or amyotrophic sclerosis, or even be absent due to amputation. Neurophysiology has proven that certain parts of the human body are represented in a topographic order on the surface of the primary sensorimotor and pre-motor cortices, the representation of which is also known as a "Homunculus" (see Figure 2). Building on these findings, the BBCI should be capable of recognizing these two spatio-temporal brain patterns as belonging to two different classes. Let us say the proximal joint flexion action, denoted as `class #1`, is implemented by the up and down or left and right movement of the entire arm, i.e. by bending it in the shoulder or elbow joint. The distal joint flexion action then, denoted as `class #2`, is implemented by the hand closure and opening movement, i.e. by finger movements.

The joint-to-class assignments of the V-Arm interface are set up such that if the `class #1` or `class #2` control signal is recognized by the BBCI system, an animation of the virtual arm flexion in the shoulder, elbow or the five finger joints is performed within the next 500-750 ms; that animation is then reversed upon repeated recognition of the same control signal class. In this setup, the user will be able to quickly adapt to the V-Arm feedback application, since it executes animated movements in a manner that is largely familiar to the user. A future vision is that a patient not capable of controlling her/his own limb (in this case an arm) could then gain control over a virtual limb in a natural way. As a result, if the patient's Central Nervous System (CNS) is still intact and capable of generating the appropriate control commands for bending and grasping, and if she/he is still able to feel or imagine an amputated limb, the BBCI system would serve as a bridge between the command emission unit and the executing extremity, bypassing all the intermediate interfacing elements such as spinal cord, peripheral nerves and muscles.

At least two kinds of assistance systems exist that can help in the final stage of signal processing and perform executive functions of the limbs.

- For users with absent limbs, prostheses can be replaced by intelligent robotic mechanisms that can be controlled by the user's brain signals encoded as emitted commands.

- Orthosis can be employed for users whose limbs are present, yet who have no control of these limbs due to a neuromuscular disease. In this case, if the executive muscles of the limb are still intact and usable to implement movements, an electro-stimulator sleeve (e.g. an arm or wrist band) is employed to process the user's brain signals encoded as control commands. The sleeve transforms control signals from the V-Arm interface

into an electrical stimulation of limb muscles, such that the imagination of a certain movement yields the execution of the desired movement in the limb.

The underlying intent of this thesis was to investigate a virtual version of such prosthesis and to define the control procedure for the orthosis. The technical realization of such a prototype remains very problematic; however, these challenges are expected to be met by experts in the field of robotics and prosthesis development and research. Please note, that in spite of finished development process of this feedback application, it has only been tested in few experiments with healthy users (mainly for the reason of verification of technical functioning). Experiments with patients are still to be conducted within the scope of future research.

Chapter IV —

SOFTWARE ENGINEERING

Our civilization runs on Software
Bjarne Stroustrup.

Getting started is the most difficult part of any new process. In Software Engineering, the first thing you need to do is understand what you are going to model and ultimately develop
Grady Booch, James Rumbaugh, and Ivar Jacobson.

This chapter will investigate the technical side of the BBCI's realization together with all of the small and large satellite applications accompanying it. It comprises a complex system of several components that need to communicate with each other and work in a synchronous way within an asynchronous system. As a result, various software engineering and modeling approaches proved invaluable during the design phase of the software development. A Machine Learning scientist or even a neurophysiologist would probably consider this chapter too technical or even boring, while a computer scientist or a software engineer, after laboring through the previous chapters (presented from a rather abstract point of view in terms of the realization and implementation of the concrete software system), would probably look forward to this chapter with enthusiasm. Nonetheless, not every single stage of the modeling process is presented and several C/C++ tricks are not disclosed, respecting the author's simultaneous role as a programmer. Extremely accurate readers are asked to forgive the omission of these highly technical details.

Firstly, the overall design of the BBCI system is discussed together with how it could be separated into self-contained modules. This is followed by basic pillars of every software engineering approach – classes of the BBCI system, their hierarchical organization and dependencies together with the main processing and interfacing objects. In section 3 (the main section of this chapter), the modules of the BBCI system are described in more detail with the

assistance of UML diagrams. An essential part of that section will address the design of feedback modules. These modules are independent software units and were developed largely in an inductive manner. In some cases they required up to a dozen development cycles that included the implementing, testing and discussing of results with neurophysiologists. Chapter IV then will then complete with a description of generic data formats of several types of files used to set up the entire system configuration so as to match the experimental requirements.

1. The Overall Design

Concordia parvae crescunt —
— discordia maximae dilabuntur.
Gaius Sallustius Crispius (86 – 34 B.C.)

The overall design of the entire BBCI system can be viewed in a more abstract manner from two levels of "magnification". Viewed from the "far away" level, the architecture resembles a conveyor belt in that the processing steps are distributed over several computers that receive a single block of source data, process it, and then pass it on to the next processing stage. This procedure will be described in the first subsection. Viewed from the "narrow" level, we magnify the main processing module of the BBCI system and recognize the parallelized processing of the acquired source data, subdivided according to the classification task into either the detection path or the determination path. Each of these paths, in turn, functions in a conveyor belt manner, which is to say, imperative. These aspects and their potential advantages and difficulties are described in the second subsection.

1.1 Distribution over several computers

The enormous amount of data to be processed in a limited time clearly requires that the tasks of the entire BBCI system are distributed over several computers that communicate via Client-Server Interfaces (CSI). Moreover, this distributed concept allows for advantageous replacement of single modules that comply with the particular communication protocols.

Figure 43 illustrates the distributed design of the BBCI system. We start egocentrically, with the volunteer user ① facing her/his own computer monitor. The visual or auditory query can be presented to the user during training sessions, or, a feedback application can be run on that computer, providing the user with the feedback animation during the application sessions. Triggered by input, or self-initiated, the user's brain generates certain spatio-temporal patterns that are sensed by the electrodes mounted into the Brain-Cap ②. The EEG information (at this stage analogous voltage changes) of all 128, or a subset, of all available channels are transferred via up to four flat cables (each with 32 wires) to the four BrainProducts® BrainVision™ amplifiers ③. These perform an A/D-conversion and transmit the acquired EEG data at a sampling rate of 5 kHz and an accuracy of 16 bits via a fiber optic cable (to avoid electromagnetic interferences) to the recorder PC ④. The recorder PC runs the BrainVision™ Recorder© software and after receiving a certain amount of data, i.e. each 40 ms, performs predefined preprocessing operations (e.g. subsampling to 1 kHz, optional high-pass, low-pass, band-pass or notch filtering) and stores the data in raw format into the data base ⑤ for later offline analysis. Additionally, it acts as a Remote Data Access (RDA) server which allows up to ten

client connections and serves one data block of acquired EEG together with all the auxiliary information (e.g. control signals or event markers) every 40 ms. A second computer ⑥ runs a corresponding client, a part of both the BBCI trainer and the BBCI applier systems. Following the data acquisition, this performs preprocessing steps for feature selection, providing the classifiers with high-dimensional feature vectors. Finally, it creates a control command from the classifier's outputs, which is then emitted over the LAN to the feedback running computer. The BBCI processing module ⑥ acts as a "talker" for various feedback clients ⑦ which are implemented to play the role of a quiet "listener" via the UDP/IP protocol and a specific communication port.

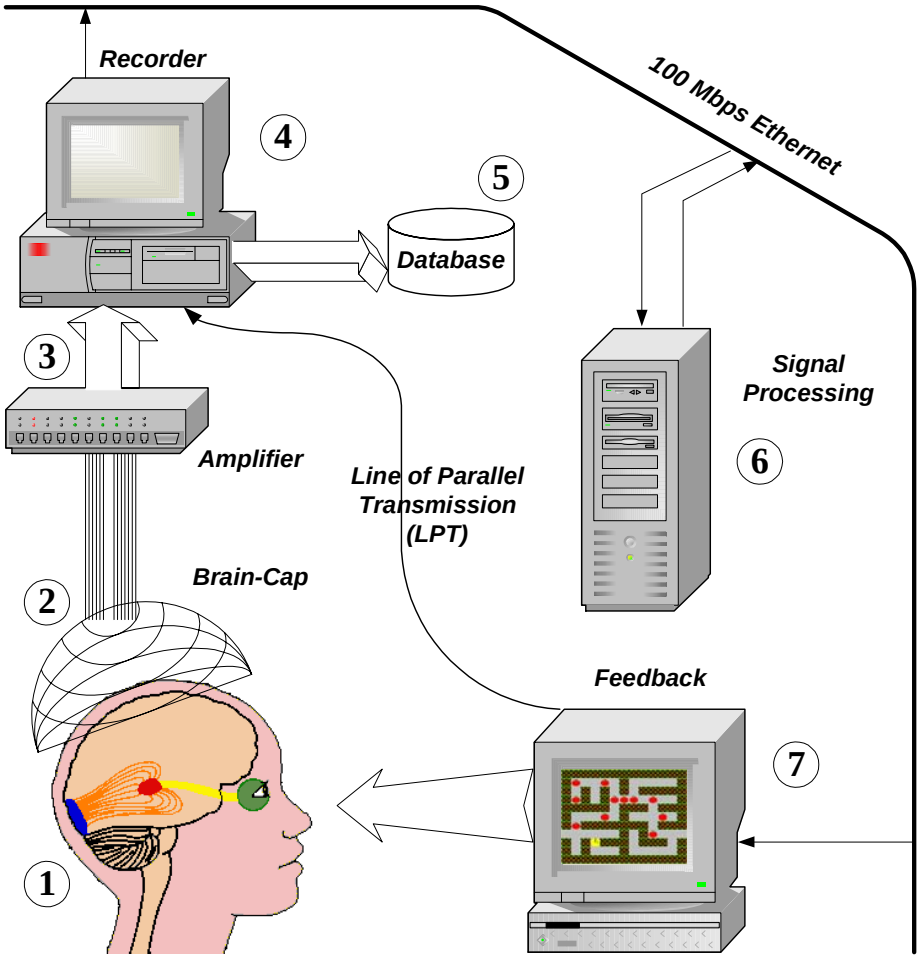


Figure 43: Distributed design of the BBCI system. The user, facing the feedback monitor, carries the Brain Cap connected to the amplifiers. The signals are transferred into the Recorder and stored in the data base. The signal packages can be transmitted over the network to the Processing PC, which transforms them ionto control commands. These commands are employed to control the feedback application that provides the user with the animation.

The feedback application now closes the loop in two ways: (*i)* indirectly, by providing the user with the animation or sound effects in order to generate certain spatiotemporal patterns in

the primary cortices of the user's brain; (*ii)* by submitting information about attaining a desired state of the animation or the performance of any animation transitions or auditory actions to the recorder PC. The latter is done by emitting coded information over the parallel port of the feedback PC. That feedback PC can, in turn, be connected to the parallel port of the recorder PC via the LPT cable. In this way, it can transmit the feedback codes to the acquired data, where the stimuli are then inserted as additional event markers, allowing the data acquired during feedback sessions to be analyzed at a later point in time in an offline or pseudo-online way. Furthermore, the complete feedback application session can then be reconstructed, i.e. the feedback animation can be replayed.

There are two main communication interfaces of the BBCI signal processing system (see Figure 43); that of the data acquisition from the recorder PC, and that of the control command emission to the feedback PC. While the first one is greatly constrained by the manufacturer of the recording system and employs the TCP/IP protocol, the latter has been developed within the scope of this work and employs the simpler User Datagram Internet Protocol (UDP/IP) communication protocol. Both protocols have advantages and shortcomings, which are summarized in Table 8.

Table 8: Comparison of TCP/IP and UDP/IP based communication protocols.

Transmission Control Protocol (TCP/IP)	**User Datagram Protocol (UDP/IP)**
Connection-Oriented	Package-Oriented
IP can distinguish between several clients and several servers connected to each other.	Establishes a "port", which allows IP to distinguish among processes running on the same host.
❑ Good control over all connected clients due to a regulated registration procedure. ❑ Reliable, even in large and complex networks, due to a well-engineered flow-control and check-sum verification. ❑ Full-Duplex capability	❑ No bothersome registration procedures needed for clients connecting to the servers. ❑ Very simple and fast implemented, although supposedly not as reliable in large and complex networks; No replies or reception acknowledgement. ❑ Easily tapped and therefore not secure.
Suitability for applications	
Require well-controlled and reliable communication in complex structured networks or WANs.	Often Connects/disconnects or reconnects, but allow transferring data insecurely. "Quick'n'Durty" implementation, e.g. prototypes.

1.1.a Communication protocol of data acquisition

The communication port number of the RDA Server is hard-coded by the manufacturer at `0xC822`. It is stored together with the TCP/IP address of the recorder PC that hosts the RDA server in the socket created to enable the communication. After establishing a TCP/IP connection, along the commonly known standard procedure, the client awaits the transmission of the

data. Each data block sent to the client conforms to the nomenclature of the RDA communication objects. Thus it is either an `RDA_MessageStart`, `RDA_MessageStop` or `RDA_MessageData`, all of which are derived from and thus contain the `RDA_MessageHeader`. This by and large defines the message type as `1`, `2` or `3` respectively. A diagram of the RDA communication objects in UML notation is illustrated in Figure 44.

The `RDA_MessageStart` is sent once each time a new client establishes the communication or each time the server starts monitoring. It contains the basic information about future data blocks, such as the number of channels and the sampling interval for identifying the dimension of the data matrix. In addition, `dResolution` helps to convert the values of the data matrix to physical values measured in µV, while `sChNames[]` is an array of textual zero-terminated strings that contain the labels of the channels transmitted (see Figure 44, left). If a connection is lost or the server stops monitoring, the `RDA_MessageStop`, which contains no own members but only those inherited from the `RDA_MessageHeader`, is sent to the client.

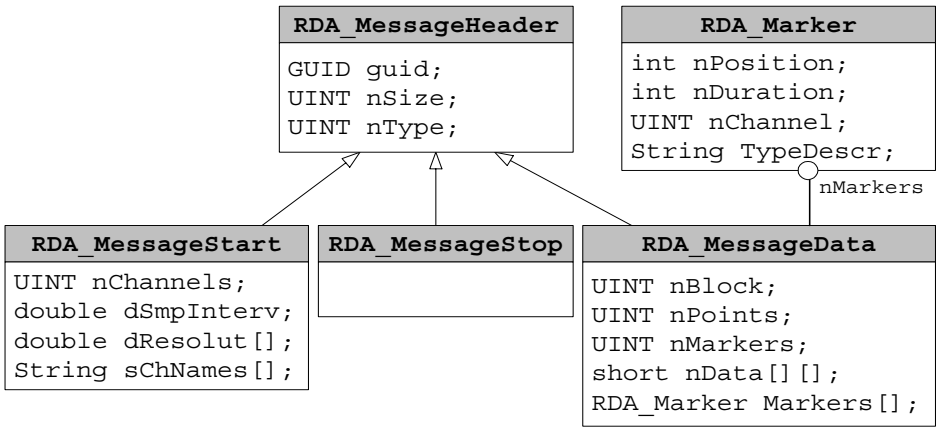


Figure 44: Class diagram of RDA communication objects. The three main classes (bottom line) are transmitted to the RDA client indicating the starting and ending point of monitoring or containing the data. They are derived from the abstract header class. The data class can also contain a set of event markers (top-right).

The acquired EEG information is transmitted in blocks of type `RDA_MessageData` containing exactly 40 ms of data each. It should be noted that in order to reconstruct a continuous data stream on the client side, it is important to provide each block with a unique identifier, for example numbering all transmitted blocks within its contents. This Block-ID is also useful for checking if data is lost; this may occur as a result of high processor load on the client side. `nPoints` reflects a value of the number of data points actually transmitted; it is redundant in most cases, since it can be calculated from the sampling interval member of `RDA_MessageStart`. It also defines the first dimension of the `nData` data array of `short` values, while its second dimension (the number of channels) is constant during the entire acquisition session as defined in `RDA_MessageStart`. Finally, the recorder software is capable of acquiring additional information, e.g. event markers or user-defined annotations that are stored in the `RDA_MessageData` as an array of `RDA_Markers`. Each marker is defined by: (*i)* its relative position, upon the event, within the current 40 ms data block, (*ii)* the number of data points to which it is relevant (usually 1), (*iii)* the channel affiliation it refers to (usually 0, referring to all channels), and (*iv)* a textual string containing its type and description as zero-terminated strings.

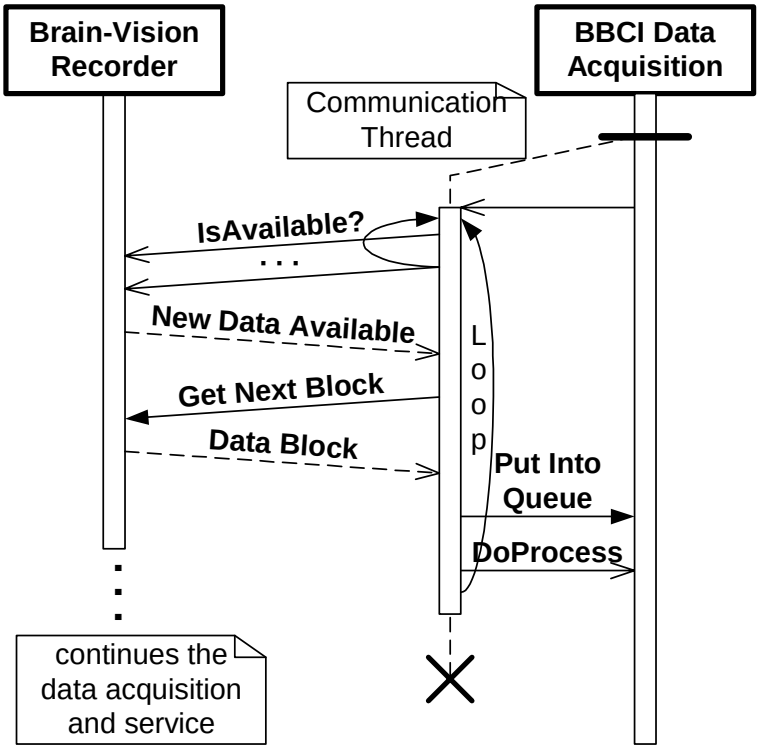


Figure 45: Timeline diagram of the data acquisition process. The BBCI system does not perform the data acquisition by itself, but its communication thread. The latter works in an endless loop waiting for, getting and storing the data package in the temporal queue; finally it emits a broadcast message. The queue can then be analyzed by other instances of the BBCI system.

The complete data acquisition process is illustrated in Figure 45 as a timeline diagram. It is important to note that the actual communication with the RDA server is performed employing a parallel thread, which renders the data processing and data acquisition as independent as possible. The BBCI data acquisition process is synchronized by its own time clock, while the synchronization of the communication thread is provided by the RDA server of the recorder PC. When an `RDA_MessageData` is received by the communication thread, its parts are queued into the continuous data stream. This is followed by a message indicating that a new processing step can be carried out, while all the new upcoming data blocks are awaited. However, BBCI data processing decides autonomously, i.e. on the basis of its own timing and other parameters, when to take care about the latest data. Usually, one data processing step is performed each time the continuous data stream is updated by the communication thread. However, after several updates, it may be possible to execute more time-consuming processing procedures. This flexible, decoupled and asynchronous design maximizes faultless data acquisition for an uninterrupted data processing.

1.1.b Communication protocol of the control command emission.

The BBCI system initiates four non-blocking threads to manage data processing, initiated by the process message of the data acquisition thread. However, before the processing starts, these threads must be initialized, i.e. filled with the appropriate semantics of the processing

procedures; this action is performed from the main BBCI thread. The data processing by itself is initiated at an appropriate point of time within the BBCI main processing thread. Data to be pre-processed is simultaneously transferred to both pre-processing threads responsible for detecting the upcoming action and determining the laterality of actions. After the two pre-processing threads indicate their readiness, the data, stored at a defined place, is passed to the two classification threads, again responsible for classifying the detection of the upcoming action and for determining the class label of the detected action's class. The main processing thread then waits once again for the ready signals of both classification threads, upon receipt of which it combines the classification results of the two classifiers, which generates a control command. This control command is then passed to a previously prepared generic feedback. The generic feedback is realized as an additional non-blocking thread, which only has the duty of communicating with the feedback application that runs on a separate computer. Thus, additional non-blocking thread serves as a network interface between the BBCI signal processing system and the feedback application. Figure 46 illustrates the data processing structure as a UML timeline diagram.

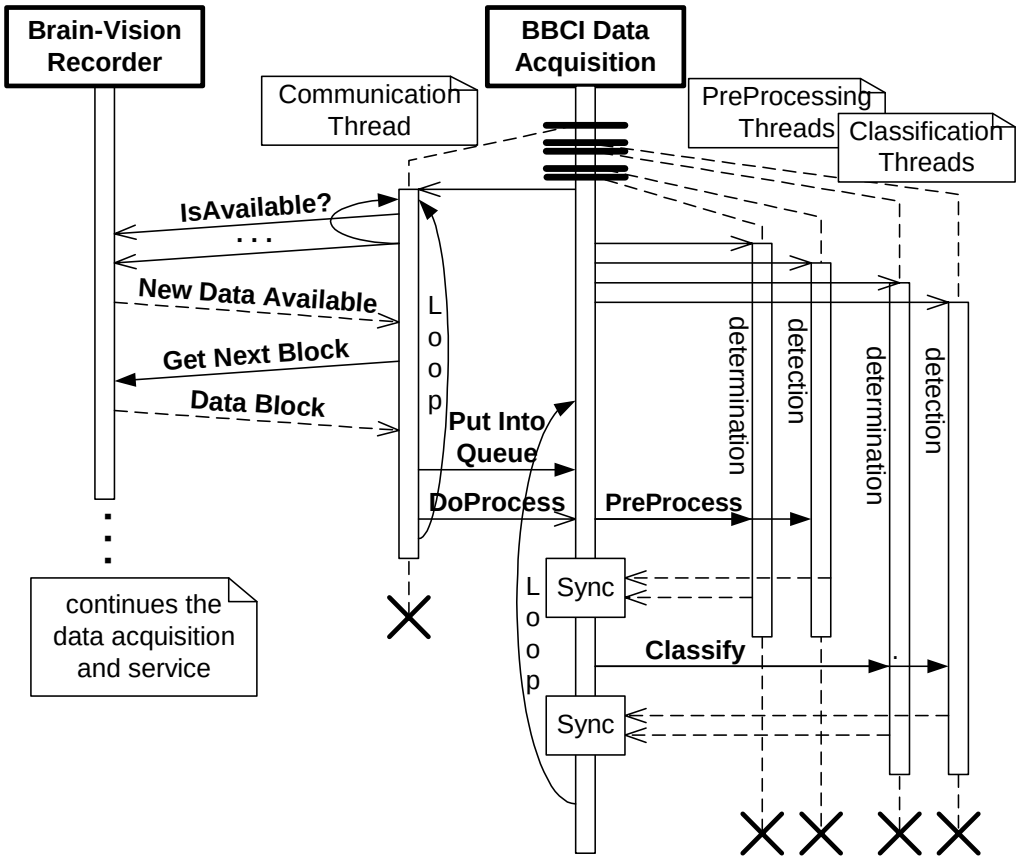


Figure 46: Timeline diagram of the signal processing and control command emission procedure. This diagram continues that in Figure 45. The data block acquired from the Recorder is analyzed by four non-blocking threads (two for the pre-processing and two for the classification). An optional synchronization step can be employed in the main thread.

The feedback communication thread carries the information concerning the host of the feedback application and the communicating port. It acts as a “talker” within a “talker-listener” constellation. The feedback application, therefore, can be realized in various ways. The simplest way consists of an infinite loop of listening to the control commands and performing the animation specified in the received control command. The feedback animation may additionally, if specified in the initialization file, indicate its actions (e.g. the reception of a particular control command) by setting a specific code onto the parallel port of its local computer. Using a parallel port connection (Line of Parallel Transmission (LPT)), it can be connected to the recorder PC and instantly place an event marker into the EEG data.

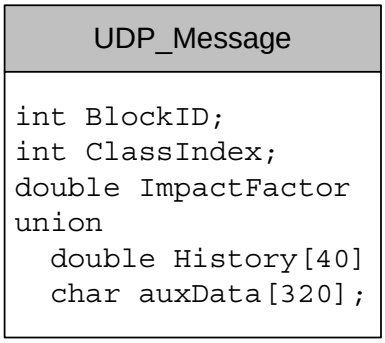


Figure 47: Class diagram of the feedback communication object that is transmitted from the BBCI system to the feedback applications. Auxiliary information about the control of the animation can be encoded within the 320 appended bytes or 40 floating point values.

A feedback communication object is defined as illustrated in Figure 47. The `BlockID` is a continuously incremented value that helps to recognize block loss. The feedback application can announce the loss without disturbing the user too much, e.g. by emitting a short `beep` message. The `classIndex` carries the information concerning the label of the most probable class of interest, and the `ImpactFactor` carries the corresponding fuzzy value. Additionally, each block includes information depending on the kind of the feedback application. Such information may include the values of the detection and the determination classifiers employed for positioning the cursor in the “Jumping Cross” feedback application, or, information on the error detection procedure employed for confirming or rejecting the last action performed in the “Brain Speller” feedback application. The semantics of the additional information space is defined in the initialization file of the generic feedback and correspondingly in the feedback application.

1.2 Partial parallelization of the classifier

As already outlined in the previous subsection, data processing of the BBCI system is performed in four non-blocking threads. These can be assigned to any individual processor; however, only the two pre-processing threads and the two classification threads can be parallelized, since the classification procedure depends on the results of pre-processing. An operating system capable of parallel data processing and running on a computer with two processing units may be instructed to assign the two detection threads to the first processing unit and the two determination threads to the second processing unit. Please note that the main processing

thread is in the `sleep` mode while the pre-processing or classification threads are at work. Figure 48 illustrates this approach.

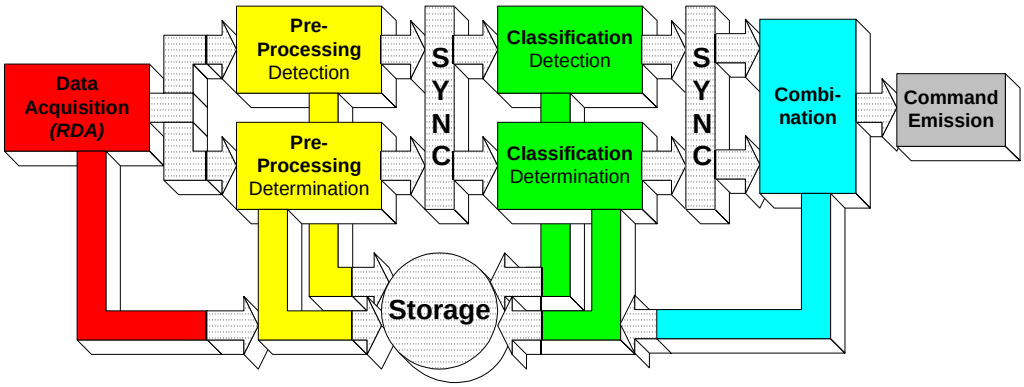


Figure 48: Parallel data processing. If the BBCI system runs on a computer with more than a single processing unit, its task can be parallelized by executing the detection and determination threads on different processors. Synchronization steps are optionally. The storage of data can be performed at all stages of the processing.

Please note also that colors denote stages of data processing, i.e. acquisition, pre-processing, classification, combination and command emission (see Figure 7, page 30). Additionally, storage of all intermediate data is possible at every processing stage. However, it is recommended that data is stored only if essential since repeated access to the hard disk drive is expensive and takes time away from the processing procedures, possibly even interrupting the synchronization or resulting in data block loss.

2. Data Abstraction

An Abstract Data Type is a data type, i.e. a set of values and a collection of operations on those values that is accessed only through an interface. A program that uses an ADT is a client, and a program that specifies the data type is its implementation.
Robert Sedgewick, Algorithms in C/C++

"The discussion of data abstraction is essentially a simultaneous discussion of two things that are highly interconnected with each other; the navigation through levels of abstraction, which all computer systems are based on, and Abstract Data Types (ADT), which in turn allow us to build programs that use different levels of abstraction, ranging from low and intermediate to high" [Sedgewick, 1998]. The following will define the main classes employed in the BBCI

system. Three types of classes are differentiated; thus, the BBCI system operates with the three types of objects[1].

- The highest level of abstraction is composed of several Graphical User Interface (GUI) objects, as is common in Microsoft Foundation Classes (MFC) based programs. These classes build the interface with the operator, govern many of the data types and manage the operation of these data types. At the very least they also serve to keep the user informed of the current state of the entire system and the processing stages of data operation.

- The operation objects make up the intermediate level of abstraction. They are defined by the specifications the operator supplied via the corresponding GUI objects. They also process data objects of the lower abstraction level.

- The data objects comprise the lower level of abstraction. Many go unnoticed by the users, who literally do not have a direct access to them. However, some selected objects also have a direct relation to the GUI objects in order to provide the operator with information about the internal state of the system. This should have been allowed only in prototype-like implementations, e.g. for result verification purposes in intermediate stages of the data processing.

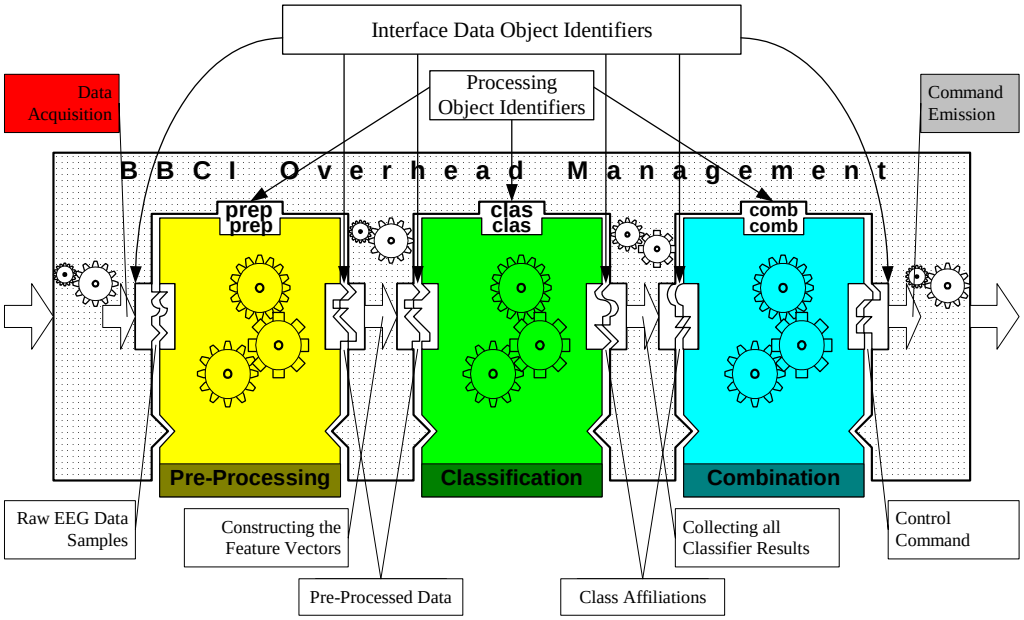


Figure 49: Modular plug-in design of the BBCI system. The BBCI Overhead Management governs the data transmission between different stages og the entire processing procedure and the synchronization inbetween. Three plug-in modules (pre-processing, classification and combination) allow for advantageous replacement and thus support fast prototyping. the data acquisition and sample selection procedures are build-in. The command emission can be realized as an additional pseudo-feedback plug-in module.

[1] Obviously there is an additional, lowest level of abstraction which, composed of standard data types or even single bits encoding information, is assumed to be trivial and a given.

Let us now discuss the role of these object types within the BBCI system. A universal BCI system must be able to employ several kinds of pre-processing, classification and combination procedures that comply with the respective communication protocols. Let us therefore encapsulate these three procedures and prescind from their modules. The remaining shell's function is then reduced to managing operations on the underlying data and processing objects. It must seek to synchronize the communication between the processing modules and verify the validity of their results. Please refer to the illustration of the modular plug-in design of the BBCI (Figure 49), where colors denote the stages of the data processing, as was defined in the previous figures.

It should be noted that the BBCI system was developed to acquire EEG data from only one type of hardware, namely BrainProducts© BrainVision™ Recorder®. This data acquisition mechanism is thus built into the BBCI system and is hard-coded. For the present prototype, that evidently constitutes a drawback that will need to be rectified, at the latest during the commercialization of the product. Yet within the scope of this project, it was not feasible to work with hardware other than the BrainProducts's© equipment and software. The same holds true for the sample selection procedure, which, while hard-coded into the BBCI overhead management module, was realized in as flexible a manner as possible so that its adaptation for most procedures still remains feasible. Finally, the control command emission is realized as a pseudo-feedback module that, equipped with the control command constructed by the combiner, communicates via a certain protocol over the network. It should be acknowledged that the BBCI system could have been illustrated as containing four plug-in modules (with an additional pseudo-feedback module). In fact, the pseudo-feedback version was the most flexible one from all feedback modules implemented. However, it would limit the BBCI to the usage of only this single feedback module; although, arguably, any other application communicating its control command through the network can be adopted to serve as the feedback application that is mastered by the pseudo-feedback module of the BBCI system. It should be mentioned that the BBCI system is able to proceed without any network-based feedback. Therefore, it can control a particular feedback application by itself, such that the command emission module is optionally included in the BBCI management overhead. Finally, the symbolic nomenclature of the above figure introduces two kinds of object identifiers.

- Processing object identifiers. These are responsible for checking that the appropriate processing steps are plugged in and executed in the correct order, i.e. that the preprocessing procedure precedes the classification, which is finalized by the combination of the classifier results, which are then passed to the feedback, if required, or to the pseudo-feedback module for control command emission. This checking mechanism is realized by a simple comparison of the identifiers of the initialization files, which are configured by the user to identify the procedures and their parameters.

- Interface data object identifiers. These are responsible for checking that (*i)* the selected procedures are capable of certain communication protocols, i.e. that the preprocessing module is able to receive the data samples selected from the raw continuous EEG encoded in a particular manner, and that (*ii)* its result (the pre-processed data) is encoded in another predefined structure so that it can be read and understood by the BBCI overhead management module. These preprocessed data structures will be acquired during the training session and, after accumulation, passed in one go to the classifier for training purposes. Nevertheless, single pre-processed data feature vectors will be passed to the classifier for classification purposes during the application session. The classifier results are then passed, in a structure containing a set of fuzzy values, to the combiner, which is capable of handling its semantics and producing an interface data object containing the control command to be emitted to the feedback application.

The following subsection describes this hierarchy of the data processing and the interaction of processing and interface data objects.

2.1 Class hierarchy

In order to understand the hierarchical class model at the core of the BBCI system, we recall its informal representation in Figure 49 and its UML notation in Figure 50. The highest level of the class hierarchy is constructed from a single object – the BBCI Overhead Management – which is a GUI type. This object contains and manages all the other processing and data interface objects that are directly or indirectly nested in it. Two kinds of objects are represented, according to the definition at the beginning of section 2. While the processing objects are illustrated as unicolored boxes indicating the processing stage, as defined above, the interface data objects govern the communication between two subsequent processing stages, i.e. between two processing objects, and are filled with two colors corresponding to the processing stages. The four main processing objects are defined as abstract classes such that they can serve as platforms for derivations of specific classes, although there are no instances of such objects. Several specific pre-processing objects, e.g. the `BBCI_FS` or `Matlab_PrePro`, are derived from the `PreProcessing` abstract class, as is done analogously with other processing objects.

Special attention must be given to the Matlab-based pre-processing and classification classes. These serve as an interface between the BBCI system and the Matlab processing engines. Constructors of these objects launch a Matlab application window and store the pointer to its top-level object, the `Engine`, into its `eEngine` member element, so that (*i)* the data to-be-processed can be put from the C++ arrays into the correct place of the corresponding Matlab environment, that (*ii)* the processing command can be called and (*iii)* the resulting data can be passed back to the appropriately prepared C++ arrays. These objects do not perform any processing by themselves and merely advise Matlab to do so in a way specified by their `doPreProcess()`, `doTrain()` and the `doApply()` member functions. The first member function executes the `sPreProCmd` string to perform the pre-processing operation with parameters that are previously set as specified in the `sParameterString`. The remaining two member functions execute the `sTrainCmd` string for initiating the training process to obtain a classifier, and the `sApplyCmd` string for the classifier application to a single selected data sample.

Let us now trace the signal's path through the BBCI system. Starting with the `EEGData` object, the `selectSamples()` member function of the `BBCI_Overhead_Management` extracts the training samples according to the markers (defined in the data as the `mrk[]` array of `BVMarkers` during the training sessions) or simply takes the latest epoch from the data (during the application session), resulting in an array of, or a single, `EEGSample`. This/These is/are forwarded to the abstract `PreProcessing` class; its virtual `doPreProcess()` member function is called. One of the specific objects of the `PreProcessing` class, which is derived from the abstract one, redefines this member function, and the correct pre-processing procedure is executed. The results of this pre-processing procedure are finally copied into the `PreProData` object defined within the `BBCI_Overhead_Management` class. Finally, the completion message of the pre-processing procedure is emitted.

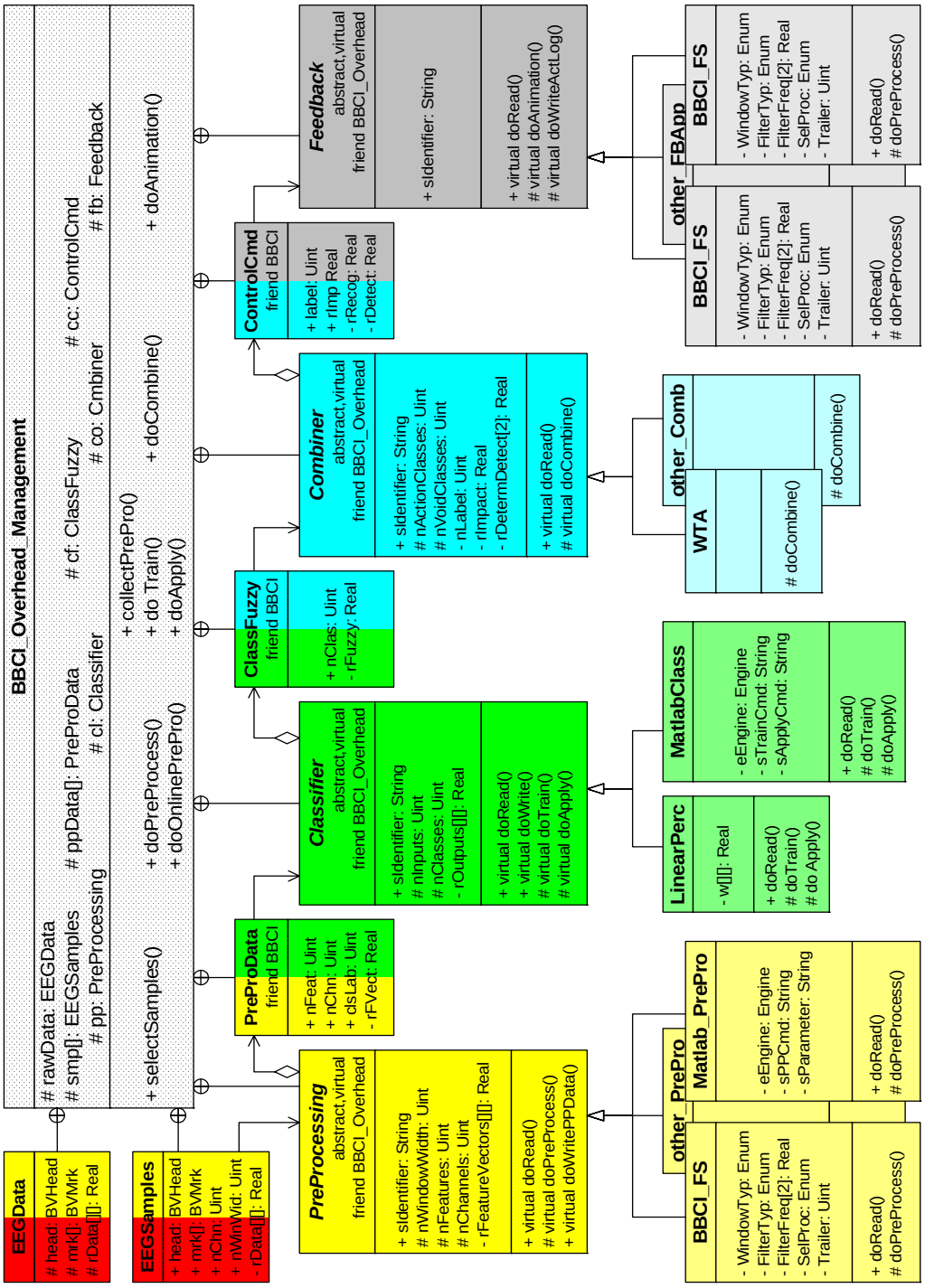


Figure 50: Hierarchical design of the data processing mechanism and the interaction between processing and interface data objects in the common UML notation for class diagrams. Unicolored classes denote processing objects, while bicolored classes denote data objects communicated through processing stages. The main BBCI Overhead Management class governing all other and the synchronization between them is pictured as while-dotted.

When both the detection and the determination pre-processing threads are completed, the `PreProData` object is transmitted to the abstract `Classifier` class; one of its virtual member functions is called. This can be either `doTrain()` or `doApply()` for the training session or the application session respectively. The feature vector(s) is/are passed to the re-implementations of these virtual member functions within the object of the specific class derived from the abstract `Classifier` class. This calls the correct classifying procedure. The classifier fills the object of the class `ClassFuzzy` previously created by the `BBCI_Overhead_Management`. After the object is filled with data, each of the classifiers emits a completion message. When both messages arrive, indicating that the classifier's fuzzy values are collected in their entirety, the resulting structure is fed into the combiner object. This triggers the virtual `doCombine()` member function of the abstract `Combiner` class. The procedure is then performed in the same manner as was described for the pre-processing and the classification procedures. Finally, the combination member function of a specific class, which is derived from the abstract one, joins the classification fuzzy values and creates the control command. The latter is passed to the abstract `Feedback` class and subsequently emitted to the object of a specific class derived from the abstract class `Feedback`.

Looking at the overall picture, we can trace the data being passed from one instance to the other. A zigzag line thus is drawn for several periods between the more abstract classes, concerned with data processing and synchronization of steps and the more specific classes that perform the mathematical calculations. Once the control command is generated from the data and passed to the feedback's communication thread, the entire system is ready to acquire new data. The hardest constraint is that each processing cycle must stay below 40 ms in duration, since otherwise data blocks may be lost, resulting in a feedback animation that is less correlated with the user's intentions.

3. Processing Modules and the GUI

This section discusses the processing modules of the BBCI system. On the whole, these modules might be seen as the GUI or overhead management modules. The entire BBCI system consists of several modules, which help the experiment conductor to acquire, load, process and restructure the EEG data, to extract the information from it, and to use this information for the control in a series of applications. These modules are subdivided into three categories: the main processing module, various auxiliary restructuring modules and finally, the feedback modules.

3.1 Main processing module

The main processing module consists of two large parts: the Trainer module and the Applier module. Both are capable of working in offline and online modes, although the offline mode is in effect set up to be pseudo-online, i.e. as if online. Consequently, both are capable of loading the EEG data files, which were previously saved on the hard disk or another storage media. The experimental setup required these two main processing modules to be implemented as independent entities. However, they must share the same objects for the user model, i.e. the parameters of the pre-processing procedures, the classifiers, and the identification of the combiner.

3.1.a “Trainer” module

The main window of the Trainer module is presented in Figure 51. Its task is to calculate the user model from the EEG data acquired directly from the user or loaded from a previously saved file. To achieve this it is subdivided into tree main parts: (*i)* data acquisition, (*ii)* feature selection and (*iii)* data processing, including pre-processing, classification and combination. There is a general “Setup” part, which is for administrative use, and an “Apply” button to switch into and operate the applier part of the main processing module.

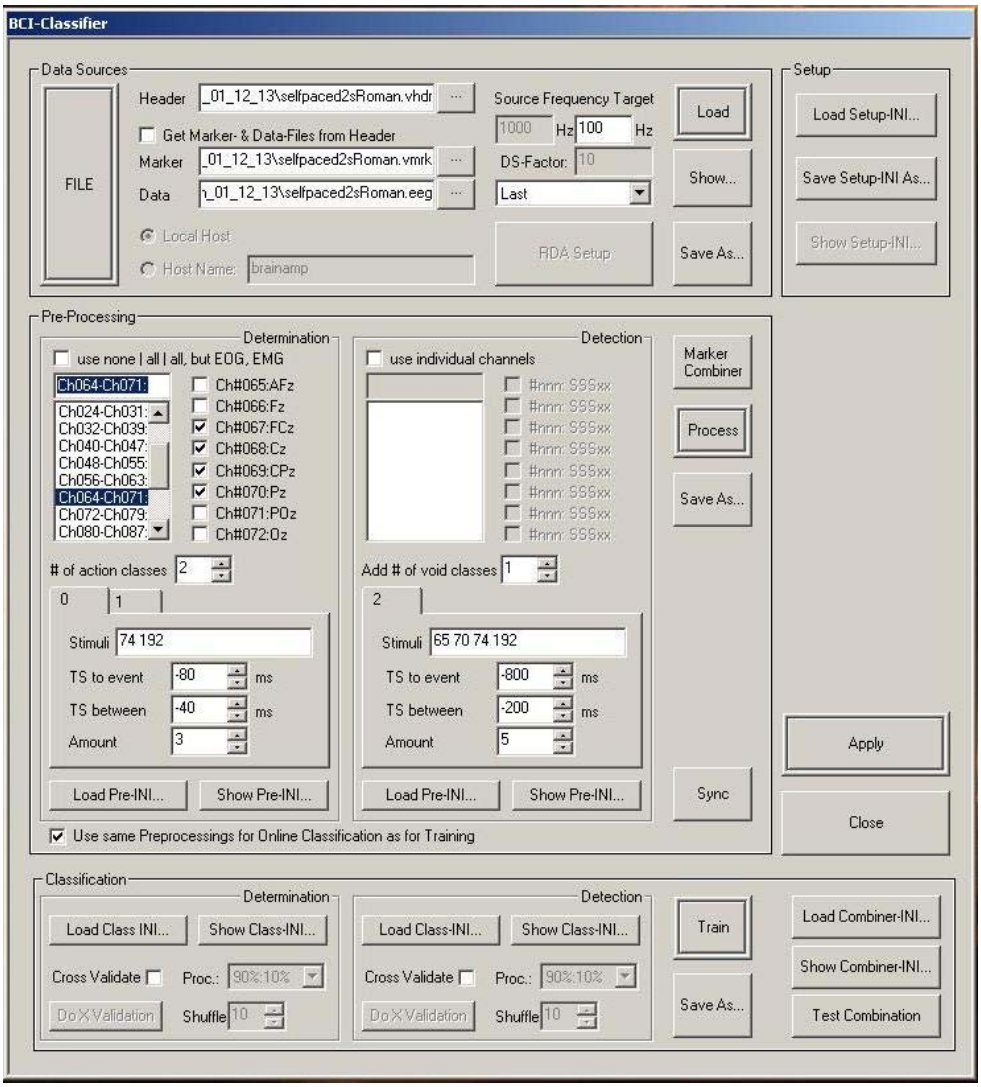


Figure 51: Main dialog GUI of the Trainer module divided into tree major parts (from top down): (*i*) data sources and acquisition, (*ii*) sample selection and preprocessing and (*iii*) classification and combination. An administrative setup part allows the experiment conductor for handling the experimental setups. The Apply button switches to the BBCI Applier module.

When starting the training session, the experiment conductor decides whether the data upon which the model will be based is already present or must be acquired. In the first case, i.e. if its label indicates “FILE”, the widgets for selecting the header file name are switched on. Alternatively, if its label indicates “RDA”, the host serving the packages can be defined, followed by the RDA-Setup, which establishes the connection and registers the BBCI Trainer as the client application. Since the EEG data of the BrainVision Recorder is stored in three interlinked files, only the filename for the header needs to be supplied; the remaining files are then recognized automatically by the links to the marker file and the data file, which are included in it. However, an option is provided to ignore those links and to issue user-defined markers and data files separately.

Often data is stored at a high sampling frequency, e.g. at 1 kHz. However, this is unnecessary for our purposes and provides too much data to be processed in real-time, which means that acquired data must be sub-sampled at a lower frequency, e.g. 100 Hz or even lower, as soon as possible. This procedure is performed automatically during data loading or acquisition, provided the value for the target frequency is selected. The Overhead Management module calculates the sub-sampling factor according to the actual data sampling rate. Several methods can be employed to sub-sample data: (*i)* calculating the mean, (*ii)* selecting the median, (*iii-vi)* selecting the last, first, middle or a random sample from the range of the consecutive, non-overlapping windows. The EEG data is then stored internally at the updated (usually lower) sampling frequency.

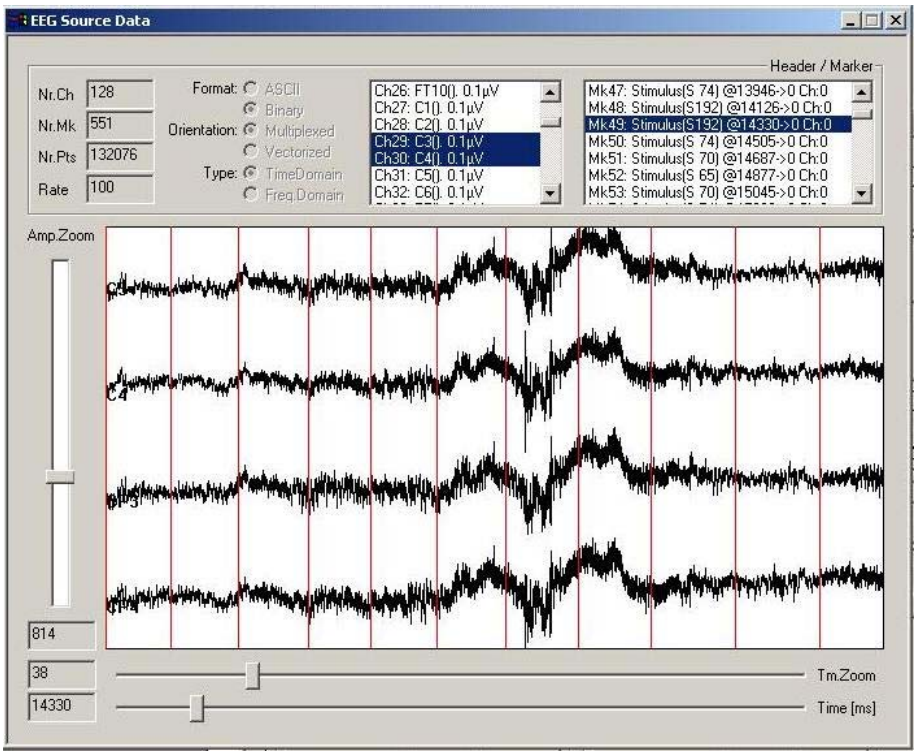


Figure 52: Inspection window of the EEG raw data. Header information is displayed statically. A subset of all available channels can be selected for display at the time point of an event marker or at an arbitrary time point. Additional scales allow magnifying the data in time and signal amplitude separately.

The loaded EEG data or the acquired and sub-sampled data can be saved back to the disk by selecting the three file names for the header, the event markers and the raw data. An important feature is the inspection of the data together with the event markers, provided in a separate window (see Figure 52). Aside from the statically displayed header information, several channels of interest can be selected and their data can be inspected. The inspection can be performed at certain points of time defined by the event markers, although an arbitrary time point can also be set by the time-slider. The amplification and temporal zoom factors can be controlled by the corresponding sliders.

Once the data is loaded into the system, all included channels are presented, in groups up to eight, in the upper combo-box of the pre-processing section. The experiment conductor is then prompted to select those whose data is to be used for the user model generation. It is common practice to select the same channels for the determination and the detection parts of the model. Therefore, initially only the determination part is active; the detection part is assumed to be identical, unless explicitly selected to be individual. In order to speed up the channel selection procedure, a tri-state check-button selects either all, none or only those whose label does not contain strings EOG and EMG. Nevertheless, it will always be possible to refine one's selection after it has been made using the check-button.

In the next step, the operator is prompted to define the number of classes of interest and to specify the parameters of the sample selection procedure for each class of interest. We distinguish here between action classes, whose samples are supposed to contain the data of the task execution, and void classes, which must only be specified if the task execution detection, in addition to the determination, is to be recognized by the user module, i.e. for asynchronous task execution experiments. The determination model will be set up based onto data samples selected from the action classes only, with each class set in opposition to all the remaining ones. The detection model will be set up based onto all data samples selected from the void classes, which are then set in opposition to both the data samples selected for each action class and to the sum of all data samples selected for all action classes. The class specification includes the *(i)* event marker labels, *(ii)* so-called stimuli (response marker labels entered as negative values) separated by a space, comma or semicolon characters, *(iii)* the time stamp t_d of the last data point (which should have been included in the sample), *(iv)* the inter-sample interval t_i (both in milliseconds) and *(iv)* the number of samples n to be extracted from the data for the current event marker. Please refer to the definition of the sample selection parameter set in equation (4) and Table 4 for appropriate values.

For complex structured data, the built-in Marker-Combiner tool will run through all the event markers and substitute combinations of two subsequent event markers (specified by certain stimuli or response values within a certain time interval) with other stimuli or response event markers. These are placed into the data at the position of the first or the second event marker's time stamp, in the center or at a random position between these two timestamps. A complex data structure exists if the intervals of tasks are given by several event markers, as is common for experiments with imagined or phantom, i.e. queried, movements, or if the synchronization is provided by external sources such as a digital metronome, while the lateralization is displayed at a prior time. (See Figure 53, left part).

The selection of the pre-processing procedure concludes the pre-processing section. Please note that you must select at least one channel before loading the pre-processing initialization file. The data format of the pre-processing initialization file is described in more detail in subsection 0. Of particular importance is the synchronous mode switch, which will launch a small dialog when in operation, asking for the synchronizing stimuli event marker labels and the time constants to switch the detection on and off relatively to the time stamp of the desired event marker (see Figure 53, right part). Please note also that a detection model is not required if the system is switched into synchronous mode, since the action's decision is performed according to the synchronizing stimuli marker. The detection part of the pre-processing and the classification sections are therefore deactivated.

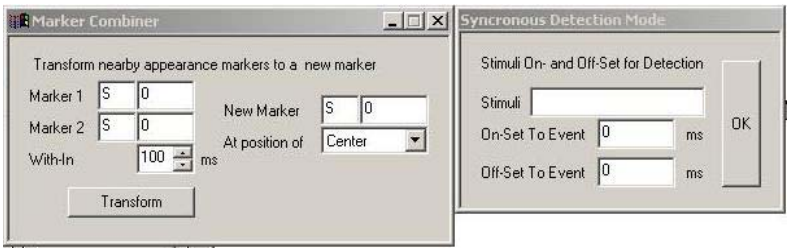


Figure 53: Dialog window of the Marker Combiner (left) and the Synchronous Mode (right) tools of the Trainer module.

When specifications for the pre-processing steps have been carried out correctly, the sample selection procedure can be run on the stored data, which will allocate enough memory for the training samples. By clicking on the "Pre-Process" button, all these training samples will be transformed to feature vectors, which can then be optionally stored as plain-text for further offline analysis. If a parametric pre-processing procedure is employed for the user model setup, the execution of two different commands is required. One command is needed to set up the pre-processing model in the training session and another command to apply the pre-processing procedure in the application session. For this purpose, the checkbox to use a different pre-processing procedure during the online session should be selected, although both pre-processing procedures share the same parameters of the user model.

The next step is the selection of the classifier user model by loading its initialization file. The file format for the classification model is described in more detail in subsection 4.3.b. The training procedure can then be started by clicking on the "Train" button in the classification section. The classifier's parameters will then be set up and the training errors for its detection and determination parts displayed, indicating the completeness of the classifier. However, it is common practice not to rely solely on training errors, which reflect the performance of the classifier on the known data. A more reliable measure of the classifier performance can be obtained by setting up the model on a subset of all the available data, e.g. 90%, and applying this model to the remaining subset, e.g. 10%. This procedure, known as cross validation, is then performed a number of times by repeatedly selecting all possible subsets as test data. Please see section 5.3 of the previous chapter and Figure 25 on page 57.

Finally, the combiner must be selected by loading its initialization file. However, since a combiner remains almost constant for most data sets and experimental setups, a default combiner is set up automatically. The entire data processing procedure can then be run to obtain the "combined" training error. The combined values are tested by applying the pre-processing, the classification and the combination procedures to data samples that were extracted from the training data. For each sample, the result of the combiner is then compared with its label.

An additional administrative "Setup" section is helpful for preparing and saving the experimental setups before the experiment starts; it can be loaded after data acquisition. A setup initialization file covers all the information that must be entered into the Trainer module's GUI. This considerably reduces the user's waiting time between the two sessions, especially for the channel selection procedure. Moreover, a setup initialization file may instruct the GUI to start the preprocessing and classifier training procedures automatically, reducing the user module setup to no more than a couple of clicks.

All parts of the user model can be replaced by other compatible parts for which appropriate initialization files exist and which can be reset based on updated parameters. However, please note that after a reset, all successive stages of the data processing procedure require resetting. For example, changing the parameters of the sample extraction procedure will induce different training samples, which may affect the pre-processing results, and therefore produce a different classification model.

3.1.b Online “Applier” module

If the user model is set up correctly and the experiment conductor is satisfied with the model setup, clicking the “Apply” button will switch to the online Applier module. The complete user model is transformed from the Trainer to the Applier module. This includes all sample selection parameters, the pre-processing specification and its parameters, if any, the classifiers, and the combiner specification. The main window of the Applier module is presented in Figure 54.

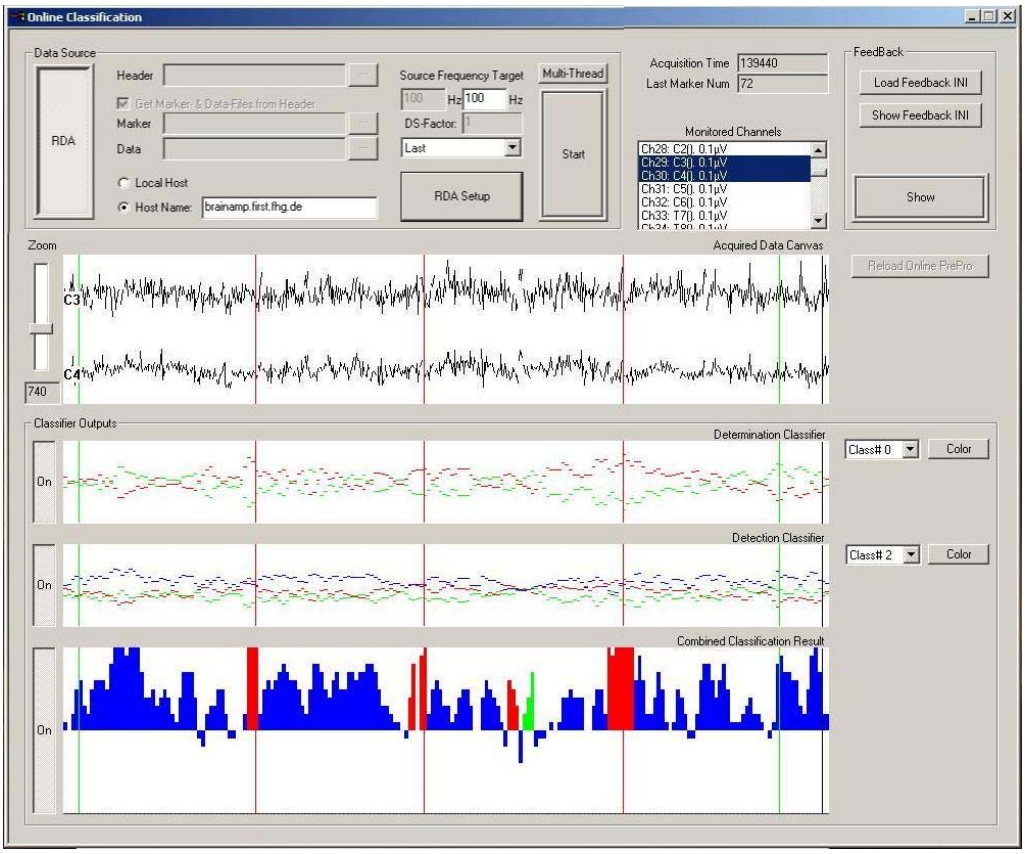


Figure 54: Main dialog GUI of the Applier module is divided into four major parts: (*i*) data source and acquisition, (*ii*) raw EEG display, (*iii*) results display of the determination, the detection paths of the classification and of the combined fuzzy values, and (*iv*) feedback selection.

The upper part of the main window of the Applier module is, in this case, almost identical with the corresponding region of the main windows of the Trainer module; it is common to switch into the RDA mode at that point. A valid host must be identified, which, runs the BrainVision Recorder switched into the server mode (RDA). The large “RDA Setup” button will establish the connection to the server and register the BBCI Applier as a client, indicating the operation’s success by the activation of the “Start” button. It is then possible to start the acquisition by clicking on this button. One step to the right are fields displaying acquisition timing properties for current data, the number of event markers currently acquired and a list box of all data channels available for display.

The remaining space is subdivided into three graphic display fields: (*i)* the presentation of the data currently acquired from the selected channels, (*ii)* the classification results for the determination and the detection classifier shown separately and (*iii)* the results of the combination. To speed up the system performance, the user may prefer to hide all graphic information, so that no data channels have to be selected, and to switch off the view showing the outputs of the fuzzy values and the combination results. The latter can be done by deselecting the on/off switches located on the left side of the corresponding display regions. The classifier outputs are given by the scaled fuzzy values of the classification results. The combiner results are displayed as histograms consisting of colored bars; their amplitudes are indicated by the value of the impact of the currently recognized class and the filling color specified for each of the classes of interest. Colors are set to default RGB values, although they can be changed using widgets to the right of the corresponding fields. Event markers are drawn as thin vertical lines at time stamps where they appear in the data.

The upper-right section lets one specify which feedback application the control commands will be sent to. If no selection is made, the data will not be sent to any application. The specification of the feedback application is realized by loading the initialization file, followed by clicking the “Show” button, which initializes the application. The initialization procedure is unique for each application. For example, for graphical feedback applications, it opens a window, and for pseudo-feedback applications such as the UDP-based feedback interface, it establishes the appropriate network connection. The file format for the specification of the feedback application is described in more detail in subsection 4.3.d.

3.2 Additional and auxiliary modules

A variety of auxiliary modules is implemented. While most of these need not be discussed in detail, two of them are briefly presented here: a simple management system for event markers and the UDP Package Manager.

3.2.a Event marker organizer

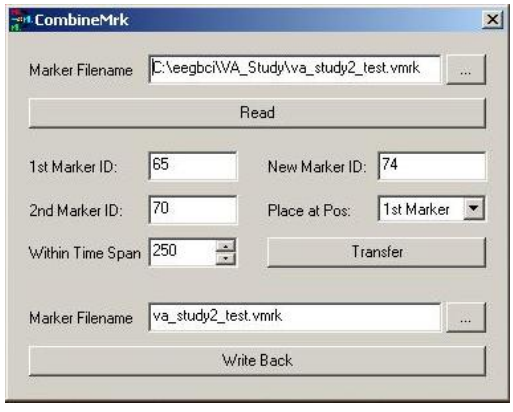


Figure 55: Main dialog window of the Event Marker Management tool. This is capable, after reading a marker file, of transforming them according to the operators demand; a new marker file is then saved, updating the old one.

As already outlined in Figure 53, there is a need to combine sequential event markers into a single one. For this purpose, a built-in marker combiner can be employed during the sample selection procedure. Because this operation needs performing every time the data is loaded, it proved useful to perform a one-time marker combination before loading the data. The event marker organizer works independently from the header and the data files, similarly to the event marker combiner tool built into the Trainer module. It then replaces the event marker file with an updated one so that the experiment conductor is no longer required to type in these values each time a new user model is set up.

3.2.b Network UDP/IP package manager

For many feedback testing purposes it proves useful not to have to set up the user module every time a new UDP-based feedback application is tested. An alternative is to set up the user module once and to perform a pseudo-online application of this model on EEG data. The pseudo-online application will then emit a series of UDP packages, which will be acquired not by the feedback application directly, but by a managing application that allows these to be stored in a particular file format. Consequently, this allows the application, after loading this file, to act as a UDP server for a wide variety of feedback applications. Figure 56 demonstrates the main dialog window of the UDP Network Package Management Tool after loading the data in "local" mode. This tool will write the currently acquired data stream into a file, provided it is switched to "sending" and "local" mode. It is capable of two file formats: (*i)* an editable plain-text format based on the Windows INI-file format, and (*ii)* the compressed binary format, suitable especially for large data streams that do not require hand editing.

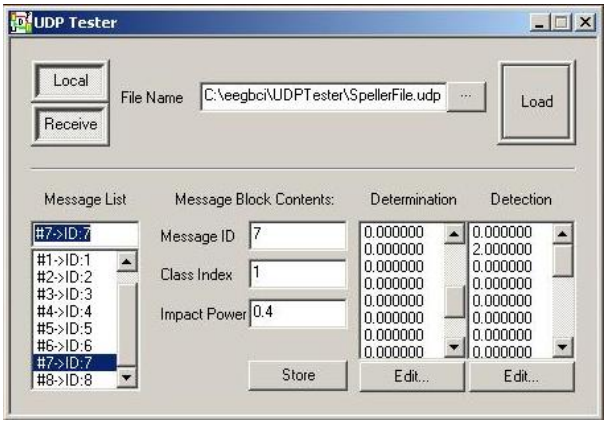


Figure 56: Main dialog window of the UDP Network Package Management tool while loading a package archive file. This tool is capable of receiving and sending a UDP package stream that can be saved or loaded in ASCII or compressed binary notation. Particular changes of the currently processed packages can be modifyied via the tool's GUI.

The tool is also capable of changing parts of the UDP stream through its GUI. As a result, after selecting a certain block, it's ID, class index and impact power can be changed. The remaining space is represented here as twenty floating point values for determination and detection, which can also be changed individually.

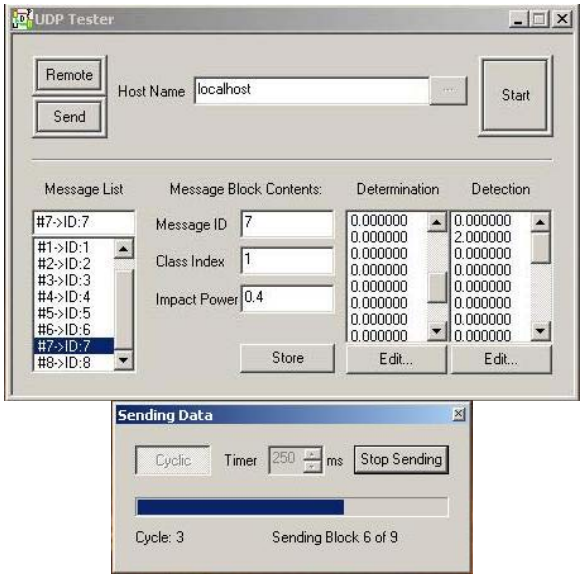


Figure 57: UDP Package Management tool while sending the data.

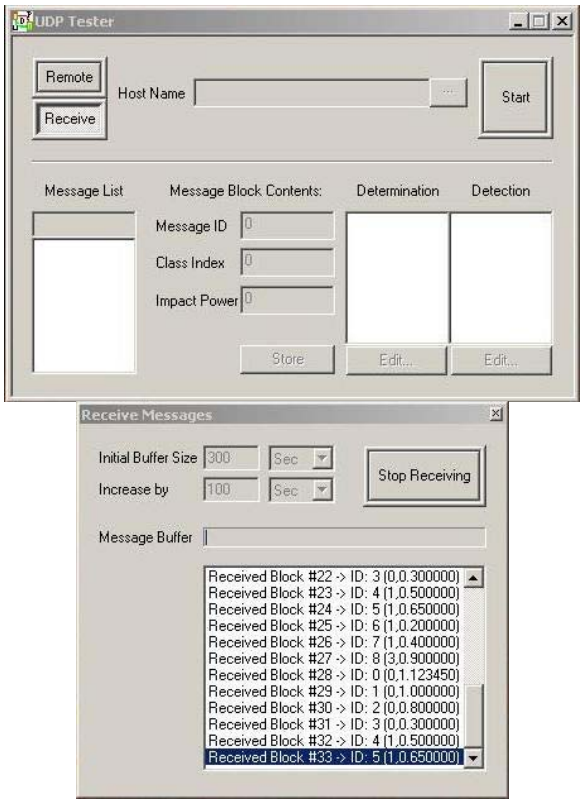


Figure 58: UDP Package Management Tool while receiving data

In the “remote” mode, the tool will send or receive the UDP packages on the communication port `0x30C9`, specifically defined for the feedback communication of the BBCI system. Figure 57 illustrates the sending procedure. The start button opens an additional dialog window that allows the sending properties to be specified; among these properties are the timing of the package emission and the automatic rewind and restart of the stream, i.e. the cyclic mode. After further confirmation, this module starts to “talk” over the network, indicating the progress by certain widgets on the bottom part of the window.

The reception of these data blocks must be initiated before starting the “talk” procedure in order to avoid data loss. For this purpose, the start button creates an additional tool window that allows setting the initial buffer size for data acquisition, i.e. how much data is expected to be acquired. The buffer will be extended automatically if the receiving procedure is not stopped before the overflow.

The confirmation click on the “Start Receiving” button will initiate a “listener” on the local machine at a specific communication port. All the data blocks are shown immediately after acquisition in the appropriate list box below. All acquired blocks will be transferred into the main tool window GUI after the acquisition is stopped.

3.3 Bio-feedback modules

Initial feedback applications have been developed as a part of the online applier, although the rising consumption of computational resources (e.g. time needed for repainting animated screens) has prompted the development of a simple pseudo-feedback application that allows the pre-processing and classification to consume as much computational power as required. A further issue that arises is the independence of resource consumption of the applier module’s processing parts from the feedback application it controls. Moreover, in future use of the BBCI system, the Overhead Management module should remain unchanged as far as is possible. Only the plug-in mathematical modules of the pre-processing, the classification and the combination should be changed. Nor should new feedback applications affect the integrity of the implementation of the Overhead Management module. These factors explain why all feedback applications are now based on remote control with a pseudo-feedback plug-in module. The Pseudo-feedback module acts as a network “talker” and is integrated into the BBCI system analogously to the pre-processing, classification and combination Plug-In modules. A complementary network “listener” is built into each autonomous feedback application, allowing them to run on a separate computer.

This section will briefly describe the software development process and present solutions to problems that occurred while designing and developing some of the most important feedback application modules.

3.3.a Feedback scenario “Jumping Cross”

The simplest feedback application has recently been re-implemented such as to be independent from the online applier module. This encompasses two different control strategies: (*i)* the classifier’s fuzzy values, omitting the combiner, are used directly to calculate the coordinates of the cross, and (*ii)* the combiner-based information concerning the recognized class, which can be employed for highlighting the target fields.

The experimental setup of this feedback application employs two classifiers in the determination part and three classifiers in the detection part. The determination classifiers are trained to distinguish between “left” actions and “right” actions. Therefore, they are identical up to the result signs, yielding the fuzzy values $\tilde{P}_{left}^{Determ}$ and $\tilde{P}_{right}^{Determ} = -\tilde{P}_{left}^{Determ}$ respectively. The

detection classifiers are trained on: “rest” vs. “left”, “rest” vs. “right”, and “rest” vs. the union of “left” and “right” samples. They yield the fuzzy values $\tilde{P}_{left}^{Detect}$, $\tilde{P}_{right}^{Detect}$ and $\tilde{P}_{void}^{Detect}$. All these values, in addition to the class label and the impact factor determined by the combiner, must be stored in the extra space of the UDP message package. The coordinates of the cross can be calculated as follows:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \tilde{P}_{right}^{Determ} - \tilde{P}_{left}^{Determ} \\ \tilde{P}_{void}^{Detect} - \max\left(\tilde{P}_{left}^{Detect}, \tilde{P}_{right}^{Detect}\right) \end{pmatrix}. \tag{21}$$

After scaling, with parameters of the maximum deflection values for *x* and *y* separately that are provided by the initialization file, the cross can be positioned on the screen.

This feedback application includes a cyclic shaped queue, i.e. a FIFO-stack, of coordinate values for painting the tail. The length of this queue is determined by the product of the number of tail knobs and the time interval of each knob. All intermediate coordinate values can be stored in this queue.

The “Brain-Pong” feedback scenario works in the same manner, but does not include the detection part of the classifiers, such that the racket is always on the move.

3.3.b Feedback scenario “Brain-Pacman” and variations

Most gaming feedback applications require, apart from a game scenario and a high-end graphical representation, more complex behavior. This can be expressed in an activity diagram in the common UML style. The Brain Pacman application incorporates a temporal command queue that is realized in a cyclic manner. The feedback application’s command acquisition thread works in a non-blocking manner and captures the incoming control commands, storing these in the queue; it overwrites the least recent commands on buffer overflow. The queue administration provides access to the last stored control command, which is then used by the application’s scenario.

The activity of the Brain Pacman feedback application is illustrated in Figure 59. It runs in an endless loop, checking the maze and performing several actions according to the result of Pacman’s position.

With each step that Pacman “hits the wall”, the user is penalized. The procedure of checking the command queue is executed in parallel to it and steers Pacman’s head to the right or left according to the extracted control command. The “Check for new commands” action state evaluates the interpretation of several subsequent control commands. If a free space, step mark or apple tile lies in front of Pacman and it is instructed to “run a step”, the field’s value is recalled and Pacman’s head is moved one step forward in a small animation, accompanied optionally by a stepping sound. The user’s reward is then calculated based on the field’s value, and the command queue is checked in parallel for new control commands. If Pacman reaches the maze exit, the simulation ends and the final reward is displayed to the user.

A variation of the Pacman gaming scenario (the so called Turnman) works in a different manner, allowing control by a single action class. Here, the Pacman object incorporates an additional timer that emits a turn command to itself at each period. Each reception of the “move forward” command resets the timer, which allows for fast movement of the object through long corridors of the maze by emitting long trains of identical control commands. A control command can reset the timer to a negative value, which allows longer reaction periods when walking in the same direction. Subsequent rotations will then be executed faster to minimize waiting periods at crossings.

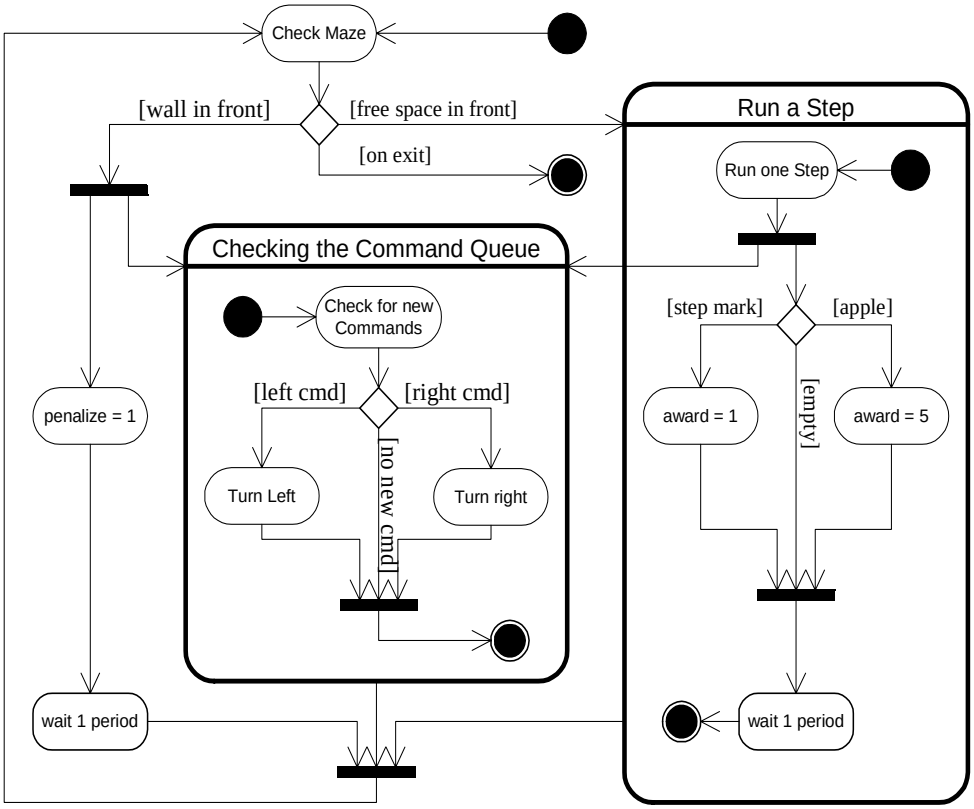


Figure 59: Activity diagram of the Pacman feedback application that can be controlled by two types of recognizable commands. Pacman runs a step if possible, while it is repeatedly checking for "turn" control commands that recently arrived on the temporal queue. In the `Run a Step` state the reward for the last step is calculated, while the command queue checker is calling Pacman's member functions that prompt it to turn.

3.3.c Feedback scenario "Brain-Tetris"

The Brain-Tetris feedback scenario works in a similar manner, with the actual brick taking the role of Pacman. Each brick contains the timer that emits the command to descend each inter-step interval. In a separate non-blocking thread, the command recognition logic runs, which analyzes the last stored control commands in the temporal queue. This analyzer emits one of the defined commands to the brick, which optionally resets the brick's internal timer. The command recognition logic requires a more complex implementation than that for the Pacman or Turnman scenarios, since it must be capable of emitting of up to four different command types: move one step left, move one step right, turn the brick by 90°, and drop the brick to the lowest possible row. Of note here is the Brain-Tetris scenario's capability to recognize further control commands for a single inter-step interval after dropping the brick. This enables the user to play faster and act with greater accuracy on intentions to position the brick under or within the construction made of previous bricks.

3.3.d Feedback scenario “Brain-Speller”

The Brain-Speller application consists of three parts. (*i)* The communication thread, which works independently from all other parts and which is responsible for lossless acquisition of the control command packages filled into the temporal cyclic queue. (*ii)* The administration of the screen picture, which is responsible for handling the current position within the alphabet tree, given by the current state of the `BitString`, and for appropriately adopting the screen picture. (*iii)* The command processing logic, which updates the `BitString` according to the control command read out from the queue. This part communicates with the temporal queue via the procedure that reads the control commands. It is also responsible for making preventive changes on the screen, awaiting the user reaction and conformably updating the `BitString`.

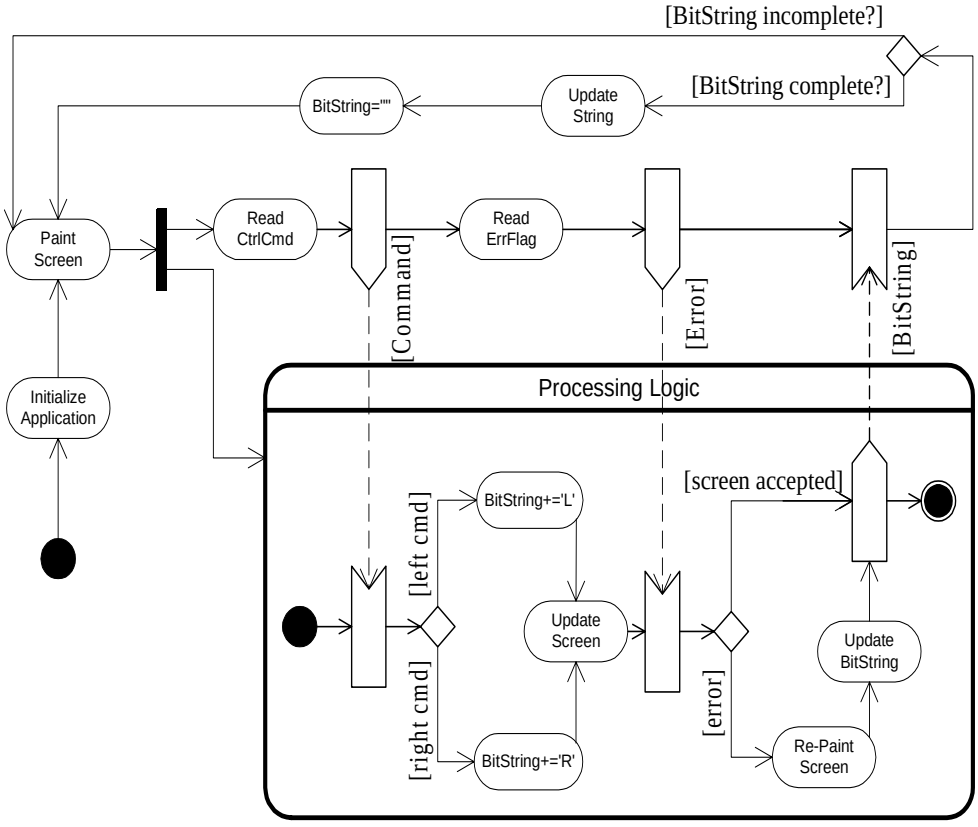


Figure 60: Activity diagram of the Brain-Speller feedback application with an automatic error detection mechanism. The processing logic waits for a new command in the queue and, after examined it, updates the user screen preventively. It waits then for the error signal and, after examined it confirms or rejects the last bit of the `BitString`. After passing it back it is examined for correspondence to a letter code followed by an appropriate update of the screen.

A new issue raised in this feedback scenario is its automatic error detection and correction capability. The performance mechanism of the Brain-Speller feedback application is illustrated as an activity diagram in the common UML notation in Figure 60. After initializing the screen, it falls into an endless loop and waits for control commands, i.e. for the signal from the UDP

package receiving thread that a new command has been acquired. After investigating the command's semantics, which detects the change in state of the animation, the command is passed to the processing logic, which updates the screen preventively to judge the user reaction. It then falls into an accept/reject cycle, when no further control commands can be acquired, until the most recent has been approved. After receiving the package containing information about the error potential, the processing logic must change the last symbol of the `BitString` (in case the user has indicated his most recent decision as an error), and transmit the updated `BitString` back to the screen administration. The `BitString` is cleared when the decision chain, reflected in the `BitString`, corresponds to the bit code of a letter. That letter is inserted into the typed string. In all cases, the screen is repainted according to the actual decision chain; an empty decision chain indicates that the application's state is currently located at the root of the alphabetic tree.

Of importance with this feedback application is user-friendliness; e.g. it must be comfortable for the user and the adaptation time must be kept as short as possible. For this purpose, it is sufficient to construct the binary alphabetic tree such that (*i)* all the letters, if the tree is being read from left to right, remain in their alphabetic order, and (*ii)* the most frequently used letters can be reached within the shortest path from the tree's root, i.e. the user will need to make fewer decisions if the letter is common and vise versa. Although these two constraints are in some sense contradictory, there is a consensual solution to this problem.

- Step 1: Let us assume a `Letter`-Object containing the symbol and its occurrence probability. Let us then collect all letters in the array `a` in the desired alphabetic order according to their symbols.
  - Step 1a: If `a` contains only a single element, then finish.
- Step 2: Find the most prominent segmentation of `a` in two halves (left and right sub-arrays `l` and `r`), so that the difference between their summed occurrence probabilities is minimized by:
  - Step 2a: Copy all letters from `a` to `r` and initialize `l` as an empty array.
  - Step 2b: Copy `r[0]` to the end of `l` and remove it from `r`.
  - Step 2c: Calculate the total occurrence probabilities of both sub-arrays $P_\Sigma(l)$ and $P_\Sigma(r)$.
  - GO TO Step 2b until $P_\Sigma(l) < P_\Sigma(r)$.
  - [1]Step 2d: IF $\left|P_\Sigma(l) - P_\Sigma(r)\right| > \left|P_\Sigma(\mathit{front}(l)) - P_\Sigma(\mathit{last}(l) + r)\right|$, THEN undo the last action performed by Step 2b.
- Step 3: Re-call procedure in Step 2 with parameter `l` as `a` in a recursive manner.
- Step 4: Re-call procedure in Step 2 with parameter `r` as `a` in a recursive manner.

---

[1] $l := \mathit{front}(l) \oplus \mathit{last}(l)$, where *last* : `Array` → `Letter` selects the last element from the array, and *front* : `Array` → `Array` yields the array containing all remaining letters, except the last one. $\oplus$: `Array` × `Letter` → `Array` is a concatenation operator appending the `Letter` to the end of `Array`.

For the English language (see probabilities of occurrences in Table 7, this algorithm will transform the alphabetically sorted array of letters into the following binary tree (see Figure 61).

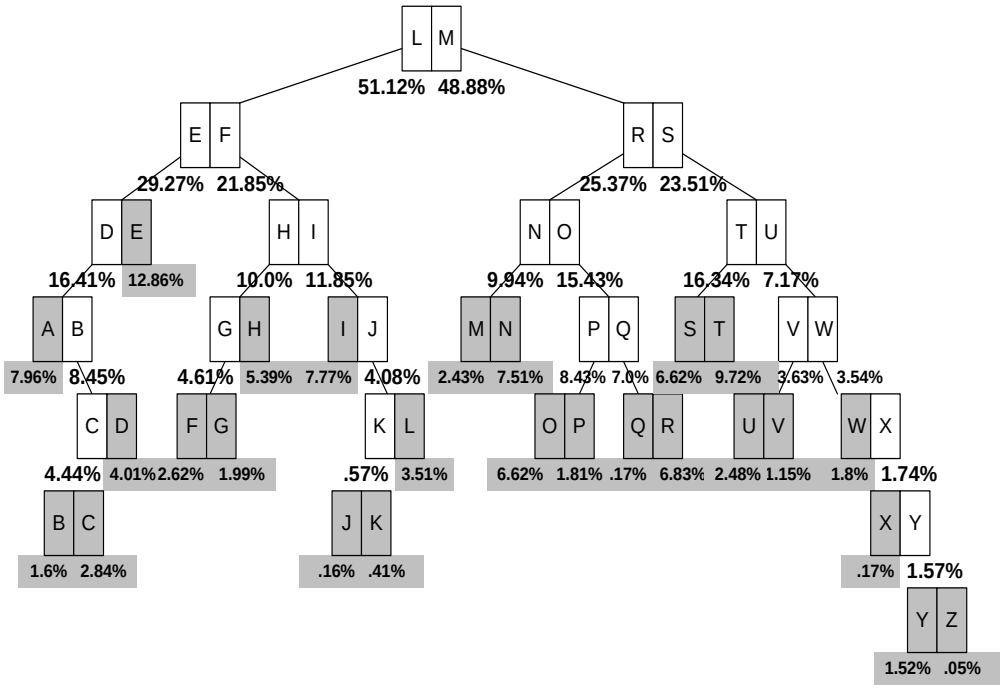


Figure 61: Alphabetic binary tree composed by the above algorithm, based on occurrence probabilities for English. The first division is made between letters ‘L’ and ‘M’, i.e. ‘A-L’ and ‘M-Z’ will be displayed on the left and right selection fields initially. For example, after choosing then the right selection field, the contents of both would change to ‘M-R’ and ‘S-Z’ respectively. Letters on grey background are leaves of the tree. Numbers below boxed enumerate occurrence probabilities of single letters of alphabet parts.

Here the two bordering letters of the most optimal division in the tree nodes are indicated, e.g. the first division is performed between letter ’L’ and ‘M’, and so on. Branches are labeled with the percentage of occurrence probabilities of sub-arrays. Letters on gray backgrounds indicate leaves of the tree, i.e. finalized selection paths. For example, to select letter ‘E’ the user has to perform the following selection sequence: “left-left-right”. ‘E’ is the most prominent letter of the English language and can thus be selected within only three decisions. ‘Z’, on the other hand, is the least prominent letter; thus, the user will require seven selection steps to select it. Please note that since all letters must retain their alphabetic order, not all paths are necessarily ordered according to the probability of the letter’s occurrence. Consequently, letter ‘Y’ is more prominent than ‘Q’, but it can be selected in six steps, thus longer than the five steps needed to select ‘Q’. For each constructed alphabet tree, an optimality criterion can be calculated that provides the average path length needed to attain the letters and according to which each decision is performed by a single bit of information required to reach the current state of the tree. The summed components (paths to letters) are weighted according to the occurrence probabilities of the letters. This optimality criterion reflects a theoretical measure of the number of bits required for the selection of a single letter. For the alphabetical

tree presented here, the user will need to submit `4.352` bits of control information to select a single letter. Although that value is not the global minimum, which is achievable without retaining the alphabetical order of all letters within the tree, it is still acceptable.

Given the occurrence probabilities, the tree calculation can be performed very fast, i.e. in logarithmic time, so that it is conceivable that the tree can be re-computed after each update of the occurrence probabilities according to the user's vocabulary.

3.3.e Feedback scenario "Virtual Arm"

The Virtual Arm feedback application, the purpose of which is described in section 7.6 of the previous chapter, employs an autonomous interface application to maintain the control commands and to remove discrepancies between the communication protocols of the BBCI online applier and the Distributed Rendering System (DRS) that manages the animation of the virtual arm.

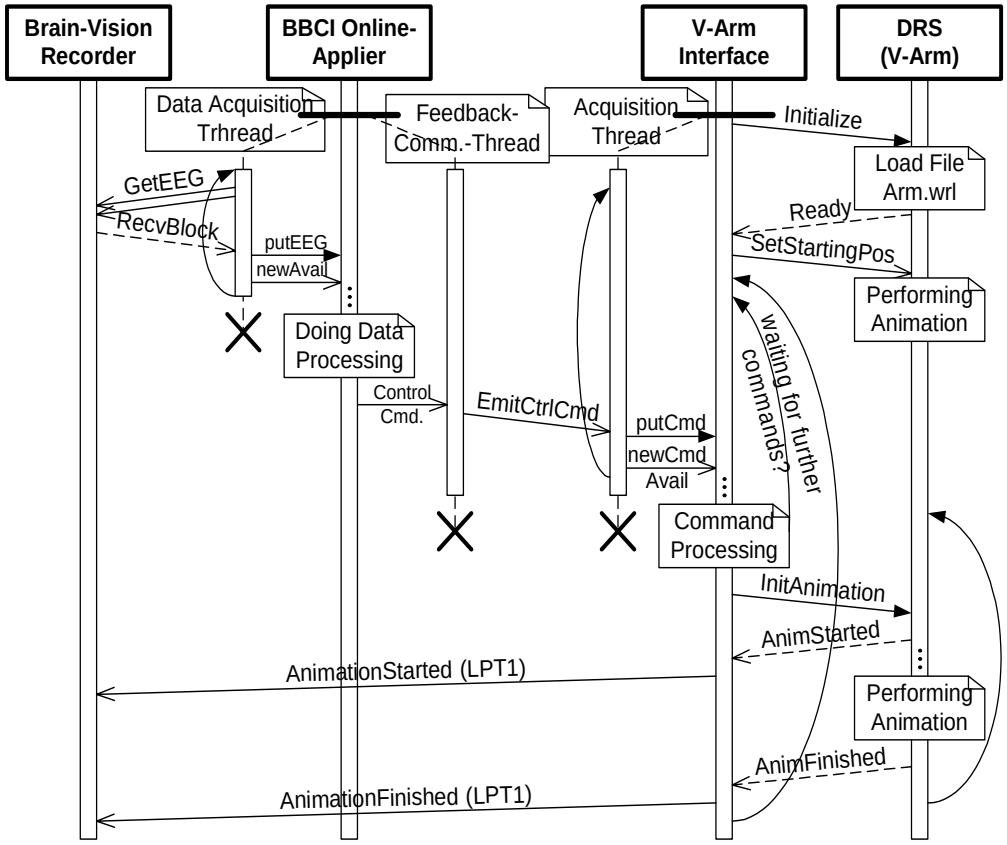


Figure 62: Control procedure of the Virtual Arm feedback application illustrated as the UML timeline diagram, building on diagrams from Figure 45 and Figure 46. In addition, the feedback communication thread emits a control command that is acquired by the V-Arm Interface's acquisition thread. After being examined, the appropriate animation is prompted to the previously initialized DRS system that can reply the animation start and finish times.

This feedback application can be controlled by the BBCI-Online Applier module in two ways: (*i)* directly via a built-in DRS pseudo-feedback, or (*ii)* via the universal UDP-based network pseudo-feedback. Since the former option is mostly trivial and straightforward (as discussed in section 1.1.b) the reader's attention will be drawn to the latter option, which additionally serves as a more general procedure. The BBCI-Online Applier module does not require reprogramming if the animation or the object handled by the DRS system changes. Only a small application serving and managing its control requires reprogramming. Figure 62 illustrates the control procedure of the Virtual Arm application via the Interface application as a UML timeline diagram (compare this with Figure 46, page 95). The role of the feedback application is assumed here by the V-Arm interface application, which, after establishing the network connection to the DRS system, initializes its animation picture by supplying the appropriate `wrl` file that contains the VR model to be animated. This also puts the arm into an initial position if needed, e.g. the V-Arm's initial and relaxing position might be slightly bent rather than fully sprawled out. The application then falls into an endless loop of waiting for a control command, analyzing its semantics and emitting the animation advice.

The communication process employed here is similar to that of the BBCI-Online Applier for receiving the raw EEG data packages. An independent and non-blocking thread is initiated, which looks for a control command message block on a certain network communication port. This communication is based on the UDP protocol. After a control command has been received, it is placed into the processing queue, also shared by the main thread, and a message is emitted indicating that new control commands are in the queue. The control command's semantics, i.e. its affiliation to one or several joint actions, is investigated and one or several animations are initiated with the appropriate animation implementation parameters. These animations require only a one-time initiating and are performed together. This does not occur immediately but over a duration within which the whole animation is to be performed, therefore generally dictating the speed of animation. Several such animations can be initiated at the same time. Upon being initiated, all animations, whether initiated jointly or individually, are stored in the DRS system and erased after completion. The performance of all the animations is then handled by the DRS system automatically. Moreover, the system indicates the starting and finishing point of the animation performance by responding messages to the calling procedure, i.e. the V-Arm Interface application.

For later offline analysis of data collected from feedback sessions, it may be of interest to know exactly when the presentation of the animation to the user started and finished. For example, it may prove useful to investigate this data in future reaction time experiments. For this purpose, a stimulus marker can be placed into the raw EEG data at the animation on/off time points. This is performed by the V-Arm interface application, by placing some values to the Line of Parallel Transmission (LPT) port of the hosting PC. If the host of the V-Arm Interface application and that of the BrainVision Recorder are coupled by a parallel line cable, it results in a stimulus marker.

The Virtual Arm feedback application was implemented and properly tested for its technical correctness in several real experiments with healthy subjects. However, there remains a need to employ the application in more challenging experiments, in order to learn from situations where the BBCI user is confronted with a complex feedback environment. This will form a central point of focus for further, wide-ranging psychological investigations.

4. Generic Data Formats

The BBCI system presented here uses a variety of file formats for storing and reading the data and meta-model parameters. These data formats are described in further detail in the following subsections.

4.1 Windows INI-files

Most of the data formats are based on the Windows initialization file format, which is defined in a canonical order. The Windows initialization file format consists of a certain identification string, which ensures that stored parameters belong to the correct semantics. Additional information may be stored in the file as a comment that must start with ‘`;`’ in its line; a comment is finished by the `EOL` (End-Of-Line) character. In the following, the file is divided into sections, each of which is headed by its own title. These titles are enclosed in brackets, e.g. `[Section Title]`, and placed on a separate line. Each section may include several keys, headed by the key name. Several types of keys are allowed:

- Binary (existence). This key allows checking whether or not a desired key is present in the file. Example: `ShowFixationCross`
- Numeral (integer or floating point valued). The key is followed by the ‘`=`’ character and the number assigned to this key. Example: `CommandActivationTerm=6`
- Textual (any arbitrary string). The key name is followed by the ‘`=`’ character and the string, which must not contain spaces, punctuation or special characters. Example: `LeftTargetActiveColor=FF8080`

The Windows Software Development Kit (SDK) and the Microsoft Foundation Classes (MFC) provide appropriate functions and parsers for maintaining these initialization files, such as `{Get|Write}PrivateProfile{Int|String}(…)`.

4.2 BrainVision’s data format

The data acquired by the BrainVision Recorder software is stored in three types of files: acquisition headers (`*.vhdr`), event markers (`*.vmrk`) and raw EEG data (`*.eeg`).

The header line of the BrainVision data header file must consist of the following: “`Brain Vision Data Exchange Header File Version 1.0`”. All section and corresponding key names that can be included in the header file are provided in Table 9.

The header line of the BrainVision marker file must consist of the following: “`Brain Vision Data Exchange Marker File, Version 1.0`”. The file format for the marker file is summarized in Table 10.

The format of the data file, which contains the raw EEG data, is set up in the header file. However, the most common format is the binary multiplexed version of the time domain data values, each coded as a 16-bit integer. Consequently, the raw data file contains a long array of `short` typed integer values, where all channel values of the first data point are stored, followed by all channel values of the next data points. This large matrix can be easily read into the 2D array in one step using the standard C function `fread()`.

Table 9: File format of the BrainVision header file (*.vhdr).

[Section Name] KeyName	Description	Values (default)
[Common Infos] DataFile	The name of the EEG data file, containing raw data that matches the current acquisition header.	None is to be provided. Relative or absolute path to the file.
MarkerFile	Optional event marker file name.	("$b.vmrk"), Relative or absolute
DataFormat	Format of the EEG data file.	ASCII, (BINARY)
DataOrientation	Storage type of the data matrix, indicating whether the storage is channel-wise or time-point-wise.	(MULTIPLEXED), VECTORIZED
DataType	Type of data stored in the EEG file, indicating temporal data or frequency amplitudes.	(TIMEDOMAIN), FREQUENCYDOMAIN
NumberOfChannels	Number of available data channels.	Integer must be provided.
SamplingInterval	Sampling Interval, in µs for time domain data, in Hz for frequency domain data.	Integer must be provided.
DataPoints	Number of data points available in the data. 0 indicates all available in the data file.	(0)
[ASCII Infos] DecimalSymbol	Only necessary if DataFormat == ASCII. Symbol of the decimal division for floating point values in the data file.	('.'),','
SkipLines	Number of lines in the data file to be skipped while reading.	(0)
SkipColumns	Number of columns in the data file to be skipped while reading.	(0)
[Binary Infos] BinaryFormat	The binary format of values. Only necessary if DataFormat is BINARY.	IEEE_FLOAT_32, (INT_16), UINT_16
UseBigEndianOrder	Indicates the use of the BigEndian binary order of the integer format data values. Only necessary if BinaryFormat is not IEEE_FLOAT_32.	(NO), YES
[Channel Infos] Ch<X> X enumerates the number of the channel, starting at 1	The list of all available channels. The values are composed of a comma-separated list: <Channel Name>,<Reference Channel Name>,<Resolution>	For example: Ch1=C3,Ref,0.1

Table 10: File format of the BrainVision marker file (`*.vmrk`).

`[Section Name]` `KeyName`	Description	Values
`[Common Infos]` `DataFile`	The name of the EEG data file, containing raw data that matches the current event marker file.	optional (“`$b.eeg`”)
`[Marker Infos]` `Mk<X>` `X` enumerates the number of the marker, starting at 1	The list of all event markers, where all marker features are comma-separated: `<type>,<description>, <position>,<duration>, <channel nr>` [,`<date/time>`]	For example: `Mk1=Time_0,,0,1,0` `Mk2=Stimulus,S2,24,1,0` `Mk3=Response,R2,38,1,0`

4.3 “Trainer” setup

The Trainer module of the BBCI system allows the experiment conductor to save the GUI settings required to reproduce the experiments. Consequently, it is possible to prepare the experiments beforehand and to load all setup parameters when required in a single step. This will reduce the user’s waiting time after the training data has been acquired and before the training procedure can start. The processing steps of the training procedure can also be initiated automatically, as all the necessary parameterization becomes available during the load process of the trainer setup initialization file.

The header line of the trainer initialization setup file must consist of the following: “`FhG-FIRST IDA Experimental Setup Definition File Version 1.0`”. All section and corresponding key names that can be included in the setup initialization file are provided in Table 11.

Please note that it is possible to load the trainer setup file in any state of the Trainer module, although the preprocessing procedure will not be executed if no training data has been loaded or previously acquired. Furthermore, training of the classifiers will not be initiated automatically if the pre-processing procedure has not been completed beforehand. If no filename was given, a default combiner will be loaded automatically. In addition, loading an updated setup file does not affect the training data and the pre-processed data currently stored in the local memory of the system. The setup loading procedure aborts with an error message if invalid filenames are given for the pre-processing, classification or combination initializations.

Table 11: File format of the BBCI Trainer setup file (`*.set`).

[Section Name] KeyName	Description	Values (default)
[Channel Usage] Determination	A comma-separated list of channel names, whose data must be involved in the pre-processing and training procedures of the determination path.	(All) | NoArte | <Ch1>[[,<Ch2>]…] Example: C3,C4,CP3,CP4,Cz
Detection	See above, except for the detection path.	See above, but in addition, (Same) reproduces the same channel list as that for the determination path.
[Class Definition] NrAction	Number of action classes assumed for the experiment.	Integer value must be provided.
NrVoid	Number of void classes assumed for the experiment.	Integer value must be provided.
Class<X> X enumerates the number of the class, starting at 0	Parameters of the sample selection parameter set (SSP), see equation (4). <t_d>,<t_i>,<n>,<Marker.*Label*>[,…]	For example: Class0=80,40,3,65
[PreProcessing] Determination	Localization of the pre-processing initialization file to be loaded for the determination path.	Absolute path to a *.bpp file must be provided.
Detection	See above, except for the detection path.	See above.
Process	Execute the pre-processing procedure automatically after setup.	(No), Yes
[Classification] Determination	Localization of the classifier initialization file to be loaded for the determination path.	Absolute path to a *.bcl file must be provided.
Detection	See above, except for the detection path.	See above.
Train	Execute the training procedure automatically after setup.	(No), Yes
Combiner	Localization of the combiner initialization file to be loaded.	Absolute path to a *.bco file must be provided, or empty.

4.3.a Preprocessing

The pre-processing initialization file determines which kind of pre-processing procedure will be executed on the data samples extracted from the loaded or acquired data. Furthermore, it defines the parameters of the desired pre-processing procedure. The header line of the pre-processing initialization file must consist of the following: "`FhG-FIRST IDA Preprocessing Definition File Version 1.0`". All section and corresponding key names possible that can be included in the pre-processing initialization file are provided in Table 12.

Table 12: File format of the BBCI pre-processing initialization file (`*.bpp`).

`[Section Name]` `KeyName`	Description	Values (default)
`[Common Infos]` `PPProcedure`	Name of the pre-processing procedure to be executed. Parameters of the desired pre-processing procedure will then be read from the section which is identically named.	`BBFeatureSelection`, `MatlabPP`
`Features`	Number of features to be extracted from each data sample. Only if `FeaturesPerChannel` key is not provided.	Integer value must be provided or see `FeaturesPerChannel` key.
`FeaturesPerChannel`	Number of features to be extracted from each data sample and each channel. Only if `Features` key is not provided.	Integer value must be provided or see description of the `Features` key.
`[BBFeatureSelection]` `WindowWidth`	Number of data points contained in the window, which is passed to the pre-processing procedure.	Integer value must be provided. (e.g. 128)
`WindowFunction`	Windowing function to be used for emphasizing the most probable occurrence of the information within the data sample.	`(Cos)`, `HalfGauss`, `Linear`, `None`
`Filter`	Filter kind to be applied during the pre-processing procedure.	`Notch`, `LowPass`, `HighPass`, `(BandPass)`
`LowFreq`	Low frequency of filtering.	0.0 [Hz]
`HighFreq`	High frequency of filtering. Only if `Filter = BandPass` or `Filter = Notch`	0.0 [Hz]

`Trailer`	Percentage of the data sample at the end, from which the feature components will be selected.	Integer value must be provided.
`FeatureSelection`	Type of feature selection procedure.	(`Equidistant`), `Linear`
`[MatlabPP] WindowWidth`	See above: `[BBFeatureSelection] WindowWidth`	Integer value must be provided (e.g. 128).
`ParameterFileName`	Matlab script file name, which will be executed before the preprocessing procedure starts. This should set the parameters needed during preprocessing to appropriate values.	Absolute or relative path to a Matlab Script file (`*.m`).
`ScriptFileName`	Matlab script file name, which contains the code of the preprocessing procedure.	Absolute or relative path to a Matlab Script file (`*.m`).

It is notable that in the case of a Matlab-based pre-processing procedure, a Matlab engine is created while loading the pre-processing initialization file, which evokes a new Matlab window on the desktop, which then appears minimized on the Windows taskbar. For the experiments where the pre-processing determination and detection procedures are to be executed in a parallel manner, it is not possible to stipulate that the detection path of the pre-processing procedure is based on Matlab, since it does not support inter-thread communication (the pre-processing procedure for the determination path is always executed within the main thread).

4.3.b Classification

The classifier initialization file determines which kind of classification procedure will be executed on the pre-processed data. The header line of the classifier initialization file must consist of the following: "`FhG-FIRST IDA Classifier Definition File Version 1.0`". All section and corresponding key names that can be included in the classifier initialization file are provided in Table 13.

The dimensionality of the input and output spaces must be provided in the classifier initialization file only if a classifier is loaded from a file, e.g. a previously trained version. These values instruct the reading procedure to parse the binary saved data correctly. If a new classifier is created and trained using the pre-processed data, these values will be automatically set to the appropriate values. The `*.bclp` file is a binary file containing a dump of the weight matrix followed by the threshold (bias value) of the linear Perceptron. The file format of the Matlab classifier parameter files is determined by the Matlab specifications and does not form part of the BBCI system.

Please note furthermore, that similarly to the issue described at the end of the previous subsection, detection classifiers cannot be based on Matlab.

Table 13: File format of the BBCI classifier initialization file (`*.bcl`).

[Section Name] KeyName	Description	Values (default)
[Common Infos] ClassificationProcedure	Name of the classification procedure to be executed. Parameters of the desired classification procedure will then be read from the section which is identically named.	LinearPerceptron, MatlabClassifier
Inputs	Number of dimensionality of the input space. If omitted, the number of features of the corresponding preprocessing procedure is copied.	(0), or provide an integer value greater or equal to the number of features of the corresponding preprocessing specification.
Classes	Number of dimensionality of the output space, i.e. the number of action classes for the determination path and total number of classes for the detection path.	(0), or specify an appropriate integer value.
[LinearPerceptron] ParameterFile	Filename containing the weights and thresholds of the linear Perceptron.	Absolute path to a binary Perceptron weights file (*.bclp).
[MatlabClassifier] TrainCommand	The Matlab command or script, which will be executed to initiate the training procedure of the classifier.	An appropriate string value must be provided. For example: train_LDA
ApplyCommand	The Matlab command or script, which will be executed to apply the data to a trained classifier.	An appropriate string value must be provided. For example: apply_separatingHyperplane
ParameterFileName	The Matlab script, which must be executed to initialize the Matlab engine before creating the classifier.	An appropriate string value must be provided (*.m), or omitted if not required.
WeightsFile	The Matlab data file, which contains the Matlab structure of a classifier, together with its weights.	Absolute or relative filename must be provided (*.m).

4.3.c Combiner

The combiner initialization file determines which kind of combination procedure will be executed on the classification fuzzy values to obtain the control command. The header line of the combiner initialization file must consist of the following: "`FhG-FIRST IDA Combiner Definition File Version 1.0`". All section and corresponding key names that can be included in the combiner initialization file are provided in Table 14.

Table 14: File format of the BBCI combiner initialization file (`*.bco`).

`[Section Name]` `KeyName`	Description	Values (default)
`[Common Infos]` `CombinationProcedure`	Name of the combination procedure to be executed.	`defaultCombiner`, `DetectDetermWTA`
`ActionClasses`	Number of action classes involved in the combination procedure.	An integer value must be provided. (`0`) will determine the appropriate value automatically.
`VoidClasses`	Number of void classes involved in the combination procedure.	An integer value must be provided. (`0`) will determine the appropriate value automatically.

It is notable that the definition of the `ActionClasses` and `VoidClasses` properties of the combiner definition is obsolete for the `defaultCombiner`, since it can be detected from the experimental setup automatically. Furthermore, if no combiner initialization file is specified by the experiment conductor, the `defaultCombiner` with appropriate properties is assumed.

4.3.d Bio-feedback

The feedback initialization file determines which kind of feedback application will be used. Furthermore, it specifies the parameters of the desired feedback application. The header line of the feedback initialization file must consist of the following: "`FhG-FIRST IDA FeedBack Definition File Version 1.0`". All section and corresponding key names that can be included in the feedback initialization file are provided in Table 15.

Please note also that the BBCI system is capable of controlling certain "built-in" feedback applications on its own. The network-based pseudo-feedback application `RemoteUDP` proved a very general purpose feedback application, and is therefore used mostly in the final stages of the development process. When a `RemoteUDP` feedback is applied, the application itself is responsible for the animation properties. For example, most developed stand-alone applications read out locally from their own initialization file and rely on their own file format. As the details of this file format are too complex, they will not be covered in this dissertation.

Table 15: File format of the BBCI feedback application initialization file (`*.bfb`).

[Section Name] KeyName	Description	Values (default)
[Common Infos] Procedure	Name of the application to be employed for the feedback.	No-Feedback LR-Selfpaced, VirtualArmFeed-back RemoteUDP
[LR-SelfPaced] | [Visualization] CrossSize	Percentage of the current screen height and width the cursor cross will span.	(20) An integer value.
CrossLineWidth	Number of pixels of the line width for the cursor cross.	(5), an integer value.
ChangeCross	Value to be set if the cursor, i.e. the cross, changes color when becoming locked.	Yes, (No)
MaxDetectionOutput	The maximum output value of the detection classifier; to be assumed at the upper part of the screen.	(3.0)
MaxDeterminationOut-put	The maximum output value of the determination classifier; to be assumed at the right border of the screen.	(4.0)
Show{Axes|Patches}	Values to be set if the static fixation cross or target fields are visible to the user.	(Yes), No (Yes), No
{Background| Cross}Color	Colors of the background and the cursor cross in common RGB notation (usage of decimal numbers is preferred).	(C8C8C8) (0000FF)
{Left|Right}Patch {Low|High}	Colors of the left/right patches, if they do not contain the cursor cross and vice versa, in the common RGB notation (usage of decimal numbers is preferred).	Colors must be provided.
History{Length| Step| Color}	Properties of the history tail: its length in number of steps, the duration of each step in number of blocks, and its color in the common RGB notation.	(6) (2) (00FF00)
[VirtualArmFeedback] Hostname	The network name of the computer hosting the DRS System and the V-Arm	(localhost)

	animation.	
LeftArm	Yes to be selected to animate left arm; otherwise right arm will be shown.	(No), Yes
CAT, CRT, CATh	Command Activation Term, Command Relaxation Term and the Command Activation Threshold for the commands based control.	(6), (12), (0.0)
TransformationSpeed	Floating point value in seconds, indicating the duration of the animation performance.	1.0
{Shoulder| Elbow| Wrist| Thumb| Index| Middle| Ring| Pinky| Rest}Class	Class indexes for each joint segment. These values are to be set to appropriate numbers corresponding to the class labels, occurring in the control commands.	(100) for all joint segments.
{Shoulder| Elbow| Wrist| Thumb| Index| Middle| Ring| Pinky| Rest}Angles	Two values of the angles for each joint segment indicating joint's rest and flexed position. Initially, all joints are set to the rest angles.	(0.0, 80.0) for all joint segments.
[RemoteUDP] HostName	The network name of the computer hosting the feedback application, i.e. the client or a listener. All the remaining animation and control parameters are then read from the applications initialization file.	(localhost)
Lock	Value to be set if the feedback application freezes upon reception of event markers. The server or talker will stop sending blocks for a certain time.	Yes, (No)

Chapter V —

CONCLUSION AND DISCUSSION

Angenehm ist am Gegenwärtigen die Tätigkeit,
am Künftigen die Hoffnung und
am Vergangenen die Erinnerung.
Aristoteles (384-322 B.C.)

This chapter aims to summarize the results presented in this dissertation. It is commonly known that meeting one challenge raises additional, sometimes even more vexing challenges. The BBCI project is still "in progress" and is possibly, by nature, never truly exhausted. While the BBCI team has successfully solved several problems, it is of fundamental importance that it remains eager to recognize and meet new challenges. Among these challenges is the further improvement of the BBCI system, to be discussed in the first section of this chapter. Finding solutions to some of these problems could prove difficult or even impossible at present and will require greater investigation. On the whole, the BBCI project, as a testing ground for scientific and technical boundaries, offers the opportunity to develop a diversity of fascinating applications.

In the second section, I will speculate on the future research and application of Brain-Computer Interfaces at large. The last section will briefly conclude this work by reiterating and summarizing its aims and objectives.

1. Perspectives and Further Improvement of BBCI

Brain-Computer Interfaces have traditionally been conceived and used in assistance systems for the disabled. See [Wolpaw et al., 1991], [Wolpaw et al., 2002], [Birbaumer et al., 1999] and many more. The BBCI system has demonstrated that Brain-Computer Interfaces also have an enormous potential for "fun", interactive applications, exemplified here as "Brain-Gaming". This admittedly requires further and deeper investigation. The field of Human-Computer Interaction (HCI) research is expanding to encompass brain signal based communication and interaction. That trend had its onset when, among other factors, the BBCI and other BCI systems introduced a new technique for reliably decoding brain signals and converting these into control commands. Currently, the two most prominent and promising paths of BBCI application are rehabilitation and gaming, although further application fields are conceivable, such as the monitoring of the mental state of a patient or vehicle driver. These paths must also be investigated for the BBCI system. The particular focus of this dissertation is to introduce several bio-feedback signals. These may or may not prove appropriate, but should allow a user who has taken a "cold-start" to explore and improve her/his individual possibilities in using the BBCI communication channel.

While most powerful BCI systems (except VEP or P300) require extensive user training, possibly lasting several months, it is a distinctive feature of the BBCI system to employ advanced digital signal processing and machine learning technology to train the computer, rather than the human subject. This means that the user can start communicating without extensive prior training. It is important that novel signal processing techniques are explored that will be capable of extracting even more relevant information from the acquired signals. Since the BBCI approach is to use the brain's built-in functions, present in the normal behavior of users rather than functions only gained with training, the adaptation procedure for the learning machine can be executed once in a session. Further analysis of user profiles (i.e. preprocessing, classification and combination parameters) across different users and sessions could help to develop a unique user profile, suitable for a wide variety of users and tasks. This finding would further minimize the required training and provide the BBCI system with an additional bonus. State-of-the-art machine learning offers many techniques of classification and regression, all of which can possibly be employed in the BBCI-system. However, it is still largely unclear why certain techniques are more appropriate for certain tasks or users than others.

Turning to the problem of so-called "unclassifiable" single trials, it is important to realize that it still remains unclear why these are present in the data and on which basis, neurological or physical, they lie. The development of a robust outlier analysis and removal algorithm is required for improving the classification performance and will significantly increase communication speed.

In addition, classification accuracy can be improved at the application level by employing the second information transmission loop, i.e. when an action is performed and its consequences are presented to the user. In that case the system waits for the user reaction and a possible error potential signal. However, the user reaction to the rapid changes in the feedback animation may have unforeseen consequences on her/his behavior. A considerable amount of theoretical research has recently been carried out in the field of online bio-feedback, although practical implications are still rare. Moreover, we can only speculate about the user's behavior when confronted with rapid error correction bio-feedback. Consequently, further practical experiments on a number of users are required in order to investigate these user reactions in detail and to ascertain to what degree this could be of help in the global scenario.

Several aspects of the BBCI system need further improvement: so far the system has employed a paradigm where the user intrinsically implements or imagines the accomplished movement, i.e. typing with the left or right index or little fingers. In ongoing research, this issue will be expanded to real assistance systems for disabled persons who have movement intensions and the respective neural correlates, but who have no means of producing an actual movement. While this kind of experiment is already underway, comprehensive results still remain insufficient.

In general, the question surrounding an ideal bio-feedback signal for BBCI will find different answers appropriate for each new application. This dissertation, however, has clearly shown that bio-feedback in a gaming scenario, such as Pacman, or in a virtual reality environment, such as the Virtual Arm, can be realized very naturally and successfully. Eventually such bio-feedback can enable the user to adapt to the classification engine and vice versa; the classification engine might find it easier to classify correctly in the course of mutual adaptation. One important issue concerns also the training procedure during feedback sessions, i.e. acquiring EEG data in an environment "contaminated" with the feedback animation and providing the learning machine with user data that closer resembles that from the real feedback application sessions.

It might prove useful to perform an initial training session without feedback and to integrate feedback control in an intermediate session. Consequently, it would then be the intermediate session that would provide the EEG data required for further fine-tuning of the model parameters. Several intermediate sessions may be carried out in order to achieve even greater classification accuracy, until no further significant improvement is observable. Finally, in a "real" feedback control session, the user model should be adapted as much as possible to the user's current behavior.

Another issue with pioneering appeal is also the thrilling possibility that since the BBCI bypasses the conduction delays from brain to muscles it could speed up the initiation of actions in competitive dual-player scenarios. However, the experimental design of this kind of application must differ to some extent from those performed previously and presented within the scope of this dissertation. The conceptual scheme, design and implementation of such competitive scenarios as a feedback application will require considerable innovation.

The global architecture of the BBCI system is currently designed to be quite static regarding the online adaptation to the user's current behavior, rendering it is actually difficult to adapt the user model during the application session. This drawback is due to the fact that certain online learning procedures of state-of-the-art machine learning technology failed or under-performed their static analogues in the design phase. However, it still remains possible to redesign the BBCI's architecture in a manner enabling it to repeatedly accept updated user models from another learning machine; for example, every couple minutes. This would imply the use of an additional computer, monitoring the current user actions and acquiring the corresponding EEG data. That computer would then calculate the user model parameters based on the user's current behavior and would, from time to time, submit these parameters to the data processing machine via an additional communication channel. This procedure could solve the problem of non-stationarity in long-term experiments, but would require the development of a procedure to merge the current and updated user model. This is a non-trivial process since most automatic learning procedures based on neural information processing methods result in "black-boxes". The problem of "looking inside the black box" is still unsolved and forms the basis of research currently underway at many scientific institutions.

Let us finally turn to the discussion of how much information we can expect to transmit in this type of new BCI channel. While various invasive technologies can admittedly achieve bitrates that are high enough for online 3D robot control [Nicolelis and Chapin, 2002], such techniques require hundreds of microelectrodes implanted into the brain's cortex. This therefore presents an inappropriate method for dealing with healthy subjects and would only suit itself for severely injured patients who have no hope of developing alternative communication channels. Furthermore, surgery is invariably associated with a high risk of causing inflamma-

tion of brain matter or of damaging healthy organs. As for non-invasive techniques, earlier studies of my colleagues have shown that in a pseudo-online idealization evaluation (i.e. data is recorded and analyzed later as if online), record bit-rates of up to 50 bits per minute are achievable [Blankertz et al., 2003]. In spelling tasks that are truly online with bio-feedback, single subjects can reach up to 2-3 letters per minute [Wolpaw et al., 1991], [Pfurtscheller et al., 1993]. A combination of the present BBCI approach with the "Brain-Speller" application would significantly increase this rate to 8-10 letters per minute. At first sight, this might still appear rather slow for a communication device, compared with other devices; for example, a computer mouse can achieve 300-350 bits of non-redundant information per minute [MacKenzie, 1991]. However, one should realize that a BCI communication channel is largely independent of other channels and offers a unique feature of ultra-fast action emissions for each single emission trial.

Finally, I would like to speculate on the direction that BBCI research and development will take in the immediate future. I estimate that it will seek to develop new and more natural feedback modi and feedback applications rather than re-developing well-known feedback applications; the reason being that the latter were designed to rely on other communication and control strategies. Moreover, the field of Human-Computer Interaction will be increasingly in demand and will be called on, in particular, to provide new techniques and communication protocols that can serve as a basis for BCI-based communication.

2. Future Visions of Brain-Computer Interfaces

Throughout the world, Brain-Computer Interfaces are the subject of rapidly developing research in Human-Computer Interaction and Machine Learning. Both fields will have a fundamental contribution to make regarding Brain-Computer Interfaces, a technological innovation that has its origins in human neurophysiology and rehabilitation engineering. There are several directions which BCI research could take in the immediate future:

- Design and development of intelligent and adaptive user interfaces for disabled persons, to provide them with standard communication possibilities, bypassing damaged communication channels. For example, the Brain-Speller and the Cursor Control application, both described in this dissertation.

- Design and development of user monitoring systems and control interfaces for persons employed as "controllers" of a variety of machines, e.g. drivers, pilots.

- Design and development of interfaces for controlling gaming applications for healthy subjects. This type of interaction still requires further experiments regarding the reaction times. In the unfortunate event that the reaction time of a BCI user is significantly higher than that of a user provided with a classical interface, the monitoring purpose of BCI systems will still be able to contribute an additional information channel to the classic human I/O modalities. This channel could transmit measurements of stress or workload or engagement of certain brain regions into task solving. This approach was referred to in this dissertation with gaming feedback applications like Brain-Pong, Pacman or Tetris.

- Design and development of intelligent control strategies in the field of prosthesis engineering. Prosthesis control is here designed and evaluated in a virtual reality environment, before the manufacturing of the actual prosthesis. It would allows the patient

(the future owner of the prosthesis) to develop a suitable control strategy for her/his prosthesis beforehand. This approach was referred to in this dissertation with the Virtual Arm application.

There are clearly many other uses that BCI technology could be applied to and that can hardly be summarized in the scope of this dissertation. It suffices to recognize, perhaps, that BCI's enormous potential for innovation will bring about considerable changes in many areas of research. In turn, BCI research itself will profit greatly from this, receiving interdisciplinary input from fields such as Human-Computer Interaction (HCI), user interface design, signal processing and machine learning, and virtual and augmentative reality. Consequently, it will need to prepare itself for these developments.

Invasive methods, still able to provide cleaner and more localized data, are thus expected to retain the upper hand over non-invasive techniques, at least in terms of ultimate result accuracy. It is also unclear whether and how invasive BCI systems will be used in brain-gaming applications, i.e. for healthy people. Hence, the current consensus is that non-invasive BCI technology will be employed mainly in the gaming industry, whereas invasive techniques will ultimately be the preferred technology for control systems that require precise control of actions.

3. Epilogue

The aim of this dissertation was to provide my personal point of view on Brain-Computer Interfaces in general and the BBCI system in particular. I would like to emphasize that neither the development of a new signal processing technique (i.e. the pre-processing technique), nor the machine learning based classifier optimization procedure, constituted the core of this work. Furthermore, my main contribution was to apply state-of-the-art software engineering techniques, such as UML, OOD/OOP, for the design of the architecture and the development of the prototype of the BBCI system.

The BBCI system presented here was designed to be distributed in order to adapt to the "online" constraints. The complex processes of data acquisition, pre-processing and classification require a high amount of temporal and spatial resources, i.e. computation time and memory, which in the present design could be successfully distributed over several computers. A "conveyor belt" like method was employed in the design of the BBCI system. A further important criterion of the system design was its applicability for research and development purposes. This encouraged the development of the modularized plug-in approach for encapsulated data processing procedures, which can be implemented in C/C++ or in Matlab. This flexibility allows for fast prototyping and high exchangeability of single components, which is of great importance during scientific trial-and-error processes where no particular and fixed processing procedure can be defined beforehand.

Moreover, the design of the communication interfaces and protocols for each of these components must be addressed. These were developed to be as flexible as possible, so that they need not be modified if new paradigms or applications are to be tested. In addition, it was designed to be adaptable to suit future requirements. While such demands are as yet unknown, adaptation can be achieved by the simple re-assignment and allocation of relevant values, which is to say, by defining where each value is to be assigned and what each value signifies.

Finally, I investigated several application groups, which proved either suitable or not suitable for control by a BCI system. These include instructive feedback applications in which users adapt to the BBCI system and develop certain control strategies in a bimodal learning mode. This means that, on the one hand, the machine learns from the user and updates its model parameters according to the current user's control strategy, but that, on the other hand,

the user updates her/his control strategy to produce signals that are more likely to be recognized by the machine. A further group of feedback applications consists of diverse gaming scenarios and their corresponding control strategies. The latter prove either suitable or not suitable for the selected gaming scenario in question. The final and most complex feedback application group is composed of rehabilitation applications. These are seen to be a prototype for future, real-world applications of the BBCI system.

The BBCI project would do well to consider intensifying its research of real-world applications, such as brain-gaming, Internet browser control, virtual reality and robotics. This would complement its current theoretical focus on neurophysiological paradigms and mathematical DSP and ML methods. In addition, it can benefit greatly from collaborating with researchers in the HCI community.

Nevertheless, I believe in and am fascinated by the potential power behind the idea of "reading men's thoughts". I have had the opportunity to encounter this "light", an experience that has strengthened my conviction in the extraordinary and successful future of Brain-Computer Interfacing.

BIBLIOGRAPHY

[Barkley, 1981]
BARKLEY R.A. 1981. "Hyperactive Children: A Handbook for Diagnosis and Treatment". New York. Guilford Press.

[Bashore and van der Molen, 1991]
BASHORE T.R. AND VAN DER MOLEN M. 1991. "Discovery of P300: A Tribute". In *Biological Physiology, 32*: 155-171.

[Beisteiner et al., 1995]
BEISTEINER R., HOLLINGER P., LINDINGER G., LANG W. AND BERTHOZ A. 1995. „Mental Representation of Movements. Brain Potentials Associated with Imagination of Hand Movements." In *Electroencephalography and Clinical Neurophysiology, 96(2)*: 183-193.

[Berger, 1929]
BERGER H. 1929. "Über das Elektroenzephalogramm des Menschen". In *Archiv Psychiatrischer Nervenkrankheiten, 87*: 527-580.

[Berger, 1930]
BERGER H. 1930. Über das Elektroenzephalogramm des Menschen II". 1930. In *Journal on Physiology and Neurology. 40*: 160-179.

[Bernstein et al., 1995]
BERNSTEIN P.S., SCHEFFERS M.K. AND COLES M.G.H. 1995. „Where did I go wrong? A Psychophysiological Analysis of Error Detection". In *Journal of Experimental Psychology: Human Perception and Performance, 21*: 1312-1322.

[Birbaumer, 1997]
BIRBAUMER N. 1997. "Slow Cortical Potentials: Their Origin, Meaning and Clinical Use". In *Brain and Behavior: Past, Present and Future*. Tilburg: Tilburg University Press. pp. 25-39.

[Birbaumer et al., 1999]
BIRBAUMER N., GHANAYIM N., HINTERBERGER T., IVERSEN I., KOTCHOUBEY B., KÜBLER A., PERLEMOUTER J., TAUB E. AND FLOR H. 1999. "A Spelling Device for the Paralyzed". In *Nature, 398*:297-298.

[Birbaumer et al., 2000]
BIRBAUMER N., KÜBLER A., GHANAYIM N., HINTERBERGER T., PERLMOUTER J., KAISER J., IVERSEN I., KOTCHOUBEY B., NEUMANN N., FLOR H. 2000. "The Thought Translation Device (TTD) for Completely Paralyzed Patients". In *IEEE Transactions on Rehabilitation Engineering, 8*: 190-192.

[Bishop, 1995]
BISHOP C.M. 1995. "Neural Networks for Pattern Recognition". Oxford University Press, NY. pp. 105-112.

[Black, 1973]
BLACK A.H. 1973. "The Operant Conditioning of the Electrical Activity of the Brain as a Method for Controlling Neural and Mental Processes". In *The Psychophysiology of Thinking.* Eds.: McGulian F.J. and Schoonover R.A. Academic Press, New York. pp. 35-68.

[Blankertz et al., 2002]
BLANKERTZ B., CURIO G., MÜLLER K.-R. 2002. „Classifying Single-trial EEG: towards Brain-Computer Interfacing". In *Advances in Neural Information Processing Systems, 14*: 157-164.

[Blankertz et al., 2002a]
BLANKERTZ B., SCHÄFER C., DORNHEGE G. AND CURIO G. 2002. „Single-trial Detection of EEG Error Potentials: A Tool for Increasing BCI Transmission Rates". In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-2002)* Aug. 27-30, 2002, Madrid, Spain. pp. 1137-1143.

[Blankertz et al., 2003]
BLANKERTZ B., DORNHEGE G., SCHÄFER C., KREPKI R., KOHLMORGEN J., MÜLLER K.-R., KUNZMANN V., LOSCH F. AND CURIO G. 2003. „Boosting Bit Rates and Error Detection for the Classification of Fast-pace Motor Commands based on Single-trial EEG Analysis". In *IEEE Transactions on Neural Systems and rehabilitation Engineering, 11(2)*: 127-131.

[Castillo and Wrobel, 2002]
CASTILLO L.-P. AND WROBEL S. 2002. "Multi-relational Active Learning for Games". In *Proceedings of the Machine Learning Workshop FGML 2002.* pp. 108-112.

[Chatrian et al., 1959]
CHATRIAN G.E., PETERSEN M.C. AND LAZARTE J.A. 1959. "The Blocking of the Rolandic Wicket Rhythm and Some Central Changes Related to Movement". In *Electroencephalography and Clinical Neurophysiology, 11*: 497-510.

[Comerchero and Polich, 1999]
COMERCHERO M.D. AND POLICH J. 1999. "P3a and P3b from Typical Auditory and Visual Stimuli". In *Electroencephalography and Clinical Neurophysiology, 110(1)*: 24-30.

[Cope and Deply, 1988]
COPE M. AND DEPLY D.T. 1988. "A System for Long-term Measurement of Cerebral Blood and Tissue Oxygenation in Newborn Infants by Near-infrared Tranillumination". In *Medical and Biological Engineering and Computation. 32*: 1457-1467.

[Courchesne et al., 1975]
COURCHESNE E., HILLYARD S. AND GALAMBOS R. 1975. "Stimulus Novelty, Task Relevance, and the Visual Evoked Potential in man". In *Electroencephalography and Clinical Neurophysiology, 39*: 131-143.

[Cui et al., 1999]
CUI R.Q., HUTTER D., LANG W. AND DEECKE L. 1999. "Neuroimage of voluntary movement: topography of the Bereitschaftspotential, a 64-channel DC current source density study". In *Neuroimage, 9*: 124-134.

[Curio et al., 2000]
CURIO G., NEULOH G., NUMMINEN J., JOUSMÄKI V. AND HARI R. 2000. "Speaking Modifies Voice-Evoked Activity in the Human Auditory Cortex". In *Human Brain Mapping, 9*: 183-191.

[Dawson, 1951]
DAWSON G.D. 1951. "A Summation Technique for Detecting Small Signals in a Large Irregular Background". In *Journal on Physiology, 115*: 2-3.

[Dawson, 1954]
DAWSON G.D. 1954. "A Summation Technique for the Detection of Small Evoked Potentials". In *Electroencephalography and Clinical Neurophysiology, 6*: 153-154.

[Decety, 1993]
DECETY J. 1993. "Analysis of Actual and Mental Movement Times in Graphic Tasks". In *Acta Psycholog, 82*: 367-372.

[Decety and Ingvar, 1990]
DECETY J. AND INGVAR D.H. 1990. "Brain Structures Participating in Mental Simulation of Motor Behavior: A Neurophysiological Interpretation". In *Acta Physiolog., 73*: 13-34.

[Decety et al., 1988]
DECETY J., PHILIPSON B. AND INGVAR D.H. 1988. "rCBF Landscapes During Motor Performance and Motor Ideation of a Graphic Gesture". In *European Archive of Psychological and Neurological Science, 238*: 33-38.

[Decety et al., 1989]
DECETY J., JEANNEROD M. AND PRABLANC C. 1989. "The Timing of Mentally Represented Actions". In *Behavioral Brain Research, 34*: 35-42.

[Deeke and Kronhuber, 1978]
DEEKE L. AND KRONHUBER H.H. 1978. "An electrical Sign of Participation of the mesial SMA cortex in human voluntary finger movements". In *Brain Research, 159*: 473-476.

[Deeke et al., 1976]
DEEKE L., GRÖZINGER B. AND KRONHUBER H.H. 1976. "Voluntary Finger Movements in Man: Cerebral Potentials and Theory". In *Biology and Cybernetics, 23*: 99-119.

[Dominey et al., 1995]
DOMINEY P., DECETY J., BROUSSOLLE E., CHAZOT G. AND JEANNEROD M. 1995. "Motor Imagery of a Lateralized Sequential Task is Asymmetrically Slowed in hemi-Parkinson's Patients". In *Neurophysiologia, 33*: 727-741.

[Donchin, 1966]
DONCHIN E. 1966. "A Multivariate Approach to the Analysis of Average Evoked Potentials". In *IEEE Transactions on Biomedical Engineering, 13*: 131-139.

[Donchin and Smith, 1970]
DONCHIN E. AND SMITH D.B. 1970. "The Contingent Negative Variation and the Late Positive Wave of the Average Evoked Potential". In *Electroencephalography and Clinical Neurophysiology: 29*: 201-203.

[Donchin et al., 1986]
DONCHIN E., KARIS D., BASHORE T.R., COLES M.G.H. AND GRATTON G. 1986. "Cognitive Psychophysiology and Human Information Processing". In *Psychophysiology: Systems, Processes, and Applications*. Eds.: M.G.H. Coles, E. Donchin, S.W. Porges. The Guilford Press, New York. pp. 244-267.

[Ebrahimi et al., 2003]
EBRAHIMI T., VESIN J.M. AND GARCIA G. 2003. “Brain-Computer Interface in Multimedia Communication”. In *IEEE Signal Processing Society – Signal Processing Magazine, 20(1)*: 14-24

[Elul, 1972]
ELUL R. 1972. “The Genesis of EEG” In *International Reviews on Neurobiology, 15*: 227-272.

[Evarts, 1996]
EVARTS E.V. 1996. “Pyramidal tract Activity Associated with Conditioned Hand Movement in Monkey”. In *Journal of Neurophysiology, 29*: 1011-1027.

[Falkenstein et al., 1990]
FALKENSTEIN M., HOHNSBEIN J., HOORMANN J. AND BLANKE L. 1990 „Effects of Errors in Choice Reaction Tasks on the ERP under Focused and Divided Attention“. In *Psychophysiological Brain Research.* Eds.: C.H.M. Brunia, A.W.K. Gaillard, A. Kok. Tilburg University Press, Tilburg. pp. 192-195.

[Falkenstein et al., 1991]
FALKENSTEIN M., HOHNSBEIN J. AND HOORMANN J. 1991. „Effects of Crossmodal Divided Attention on Late ERP Components. II. Error Processing in Choice Reaction Tasks“. In *Electroencephalography and Clinical Neurophysiology, 78:* 447-455.

[Falkenstein et al., 2000]
FALKENSTEIN M., HOHNSBEIN J. AND HOORMANN J. 2000. „Objektivierung altersabhängiger Änderungen von Beanspruchung und Ermüdung bei psychomentalen Belastungen am Bildschirmarbeitsplatz“. In *Bundesanstalt für Arbeitsschutz und Arbeitsmedizin*, Berlin.

[Falkenstein et al., 2000a]
FALKENSTEIN M., HOORMANN J., CHRIST S. AND HOHNSBEIN J. 2000. „ERP Components on Reaction Errors and their Functional Significance: A Tutorial“. In *Biological Psychology, 51*: 87-107.

[Fallinger et al., 1997]
FALLINGER A.J., MÜLLER T.J. AND STRIK W.K. 1997. “Neurophysiological Correlates of Mental Imagery in Different Sensory Modalities”. In *International Journal on Psychophysiology, 25*: 145-153.

[Farah, 1984]
FARAH F.J. 1984. “The Neurological Basis of Mental Imagery: A Componential Analysis”. In *Cognition, 18*: 245-272.

[Fisher, 1936]
FISHER R.A. 1936. “The Use of Multiple Measurements in Taxonomic Problems”. In *Annals of Eugenics 7*: 179-188. Reprinted in *Contributions to Mathematical Statistics*, John Wiley: NY. 1950.

[Förster, 1936]
FÖRSTER O. 1936. “Motorische Felder und Bahnen”. In *Handbuch der Neurologie IV*. Eds.: O. Bumke, O. Förster. Springer Verlag, Berlin.

[Freeman, 1983]
FREEMAN W.J. 1983. “The Psychological Basis of Mental Images”. In *Biological Psychiatry, 18*: 1107-1125.

[Friedman and Callimahos, 1920]
FRIEDMAN W.F. AND CALLIMAHOS D. 1920. “Military Cryptanalysis”. Part I, vol.2, Aegean Park Press, Laguna Hills, CA USA.

[Gehring et al., 1993]
GEHRING W.J., GOSS B., COLES M.G.H., MEYER D.E. AND DONCHIN E. 1993. “A Neural System for Error Detection and Compensation”. In *Psychological Science, 4*: 385-390.

[Glover et al., 1986]
GLOVER A.A., ONOFRJ M.C., GHILARDI M.F. AND BODIS-WOLLNER I. 1986. “P300-like Potentials in the Normal Monkey using Classical Conditioning and the Auditory ‘oddball’ Paradigm”. In *Electroencephalography and Clinical Neurophysiology: 65*: 231-235.

[Goldenberg et al., 1986]
GOLDENBERG G., SUESS E., PODREKA I., STEINER M., LANG W. AND DEEKE L. 1986. „TcHmpao Spect for Detection of rCBF changes caused by employment of imagery“. In *Advances in Neuroimaging.* Eds.: T. Reiner, H. Binder, E. Deisenhammer. Verlag für Medizinische Akademie, Wien.

[Green et al., 1999]
GREEN J.B., SORA E., BAILY Y., RICAMATO A. AND THATCHER R.W. 1999. “Cortical Motor Recognition after Paraplegia: an EEG Study”. In *Neurology, 53(4)*: 736-743.

[Hardwick et al., 1996]
HARDWICK A., RUSH J., FURNER S. AND SETON J. 1996. “Feeling it as well as Seeing it: Haptic Displays with Gestural HCI for Multimedia”. In *Proceedings of the York Gesture Workshop*, Mar. 96, University of York, York, GB. pp. 105-116.

[Harel et al., 2003]
HAREL D., CARMEL L. AND LANCET D. 2002. “Towards an Odor Communication System”. In *Computational Biology and Chemistry, 27*: 121-133.

[Harland et al., 2002]
Harland C.J., Clark T.D. and Prance R.J. 2002. “Remote Detection of Human Electroencephalograms Using Ultrahigh Input Impedance Electric Potential Sensors”. In *Applied Physics Letters, 81(17)*: 3284-3286.

[Höllinger et al., 1999]
HÖLLINGER P., BEISTEINER R., LANG W., LINDINGER G. AND BERTHOZ A. 1999. „Mental Representation of Movements: Brain Potentials Associated with Imagination of Eye Movements“. In *Electroencephalography and Clinical Neurophysiology, 110*: 799-805.

[Hoffman, 1986]
HOFFMAN R.E. 1986. “Verbal Hallucinations and Language Processes in Schizophrenia”. In *Behavioral Brain Science, 9*: 503-548.

[Ingvar and Philipson, 1977]
INGVAR D.H. AND PHILIPSON L. 1977. “Distribution of Cerebral Blood Flow in the Dominant Hemisphere during Motor Ideation and Motor Performance”. In *Annuals on Neurology, 2*: 230-237.

[Jöbsis, 1977]
JÖBSIS F.F. 1977. „Noninvasive, Infrared Monitoring of Cerebral and Myocardial Oxygen Sufficiency and Circulatory Parameters “ In *Science, 198(4323)*: 1264-1267.

[John, et al., 1964]
JOHN E.R., RUCHKIN D.S. AND VILLEGAS J. 1964. “Signal Analysis and Behavioral Correlates of Evoked Potentials Configurations in Cats”. In *Annual Report of New York Academy of Science, 112*: 362-420.

[Johnson, 1988]
JOHNSON R. 1988. “The Amplitude of the P300 Component of the Event-Related Potentials: Review and Synthesis”. In *Advances in Psychophysiology, vol.2*. Eds.: P.K. Ackles, J.R. Jennings, M.G.H. Coles. Jai Press Inc., Greenwich. pp. 69-137.

[Kalcher and Pfurtscheller, 1995]
KALCHER J., AND PFURTSCHELLER G. 1995. „Discrimination Between Phase-locked and Non-phase-locked Event-related EEG Activity“ In *Electroencephalography and Clinical Neurophysiology, 94*: 381-483.

[Klass and Bickford, 1957]
KLASS D. AND BICKFORD R.G. 1957. “Observations on the Rolandic Arceau Rhythm”. In *Electroencephalography and Clinical Neurophysiology, 9*: 570.

[Knight, 1996]
KNIGHT R.T. 1996. “Contribution of Human Hippocampal Region to Novelty Detection”. In *Nature, 383*: 256-259.

[Kosslyn, 1988]
KOSSLYN S.M. 1988. “Aspects of a Cognitive Neuroscience of Mental Imagery”. In *Science, 240*: 1621-1626.

[Krepki et al., 2003]
KREPKI R, BLANKERTZ B., CURIO G. AND MÜLLER K.-R. 2003. „The Berlin Brain-Computer Interface: Towards a new Communication Channel for Online Control of Multimedia Applications and Computer Games”. In *Proceedings of the 9th International Conference on Distributed Multimedia Systems (DMS’03)*, Miami, FL, USA, Sept. 24-26, pp. 101-108.

[Krepki et al., 2004]
KREPKI R, BLANKERTZ B., CURIO G. AND MÜLLER K.-R. 2004. „The Berlin Brain-Computer Interface: Towards a New Communication Channel for Online Control in Computer Games”. In *Multimedia Tools and Applications. Special Issue on Distributed Adaptation, Representation and Processing of Multimedia Information.* In Print.

[Krepki et al., 2004a]
KREPKI R, LASKOV P., CURIO G., BLANKERTZ B. AND MÜLLER K.-R. 2004. “По щучьему велению, по моему хотению: Берлинский Нейро-Компьютерный Интерфейс” („Durch Zauberei wünsch’ ich mir herbei: das Berliner Neuro-Computer Interface“). In *Nauka i Zhizn’ (Wissenschaft & Leben), 11*: In press. (In Russian)

[Kukleta and Lamarche, 2001]
KIKLETA M. AND LAMARCHE M. 2001. “Steep Early Negative Slopes can be Demonstrated in Pre-Movement Bereitschaftspotential”. In *Clinical Neurophysiology, 112*: 1642-1649.

[Lang et al., 1989]
LANG W., ZILCH O., KOSKA C., LINDINGER G. AND DEECKE L. 1989. „Negative Cortical DC Shifts preceding and accompanying simple and complex sequential movements”. In *Experiments in Brain Research, 74*:99-104.

[Lang et al., 1990]
LANG W., OBRIG H., LINDINGER G., CHEYNE D. AND DEEKE L. 1990. „Supplementary Motor Area Activation While Tapping Bimanually Different Rhythms in Musicians“. In *Experimental Brain Research, 79*: 504-514.

[Lopes da Silva, 1999]
LOPES DA SILVA F.H. 1999. “Event-Related Potentials: Methodology and Quantification”. In *Electroencephalography: Basic Principals, Clinical Applications and Related Fields*, 4th Edition Baltimore. Williams and Wilkins. Eds.: Niedermeyer, Lopes da Silva. pp. 947-957.

[MacIntyre and Feiner, 1986]
MACINTYRE B. AND FEINER S. 1996. “Future Multimedia User Interfaces”. In *Multimedia systems Journal*, 4(5): 250-268.

[MacKenzie, 1991]
MACKENZIE I.S. 1991. “Fitt’s Law as a Performance Model in Human-Computer Interaction”. In *Doctoral Dissertation.* University of Toronto. Canada.

[McFarland et al., 1998]
MCFARLAND D.J., MCCANE L.M. AND WOLPAW J.R. 1998. “EEG-Based Communication and Control: Short-Term Role of Feedback”. In *IEEE Transactions on Rehabilitation Engineering, 6(1)*: 7-11.

[Mendoza and Wichman, 1978]
MENDOZA D. AND WICHMAN H. 1978. “Inner darts. Effects of Mental Practice or Performance of Dart Throwing”. In *Perceptual Motor Skills, 47*: 1195-1199.

[Mika et al., 2001]
MIKA S., RÄTSCH G. AND MÜLLER K.-R. 2001. „A Mathematical Programming Approach to the Kernel Fisher Algorithm“. In *Advances in Neural Information Processing Systems, 13*: 591-597.

[Millán and Mouriño, 2003]
MILLÁN J.D.R. AND MOURIÑO J. 2003. “Asynchronous BCI and Local Neural Classifiers: An Overview of the Adaptive Brain Interface Project”. In *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 11(2):* 159-161.

[Müller et al., 2001]
MÜLLER K.-R., MIKA S., RÄTSCH G., TSUDA K. AND SCHÖLKOPF B. 2001. „An Introduction to Kernel-based Learning Algorithms“. In *IEEE Transactions on Neural Networks, 12(2)*: 181-201.

[Nicolelis and Chapin, 2002]
NICOLELIS M.A.L. AND CHAPIN J.K. 2002. “Controlling Robots with the Mind”. In *Scientific American, 287(4)*: 46-53.

[Numminen and Curio, 1999]
NUMMINEN J. AND CURIO G. 1999. “Differential Effects of Overt, Covert and Replayed Speech on Vowel Evoked Responses of the Human Auditory Cortex”. In *Neuroscience Letters, 272*: 29-32.

[Numminen et al., 1998]
NUMMINEN J., CURIO G., NEULOH G., JOUSMÜKI V. AND HARI R. 1998. “Speaking Modifies Utterance-Related Activity of the Human Auditory Cortex”. In *Neuroimage, 7*: p.48.

[Oken, 1989]
OKEN B.S. 1989. “Endogenous Event-Related Potentials”. In *Evoked Potentials in Clinical Medicine*. Ed. K.H. Chiappa. Raven Press, New York. pp. 563-592.

[Paivio, 1969]
PAIVIO A. 1969. “Mental Imagery in Associative Learning and Memory”. In *Psychological Reviews, 76*: 241-263.

[Pantic and Rothkrantz, 2000]
PANTIC M. AND ROTHKRANTZ L.J.M. 2000. "Automatic Analysis of Facial Expressions". In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(12)*: 1424-1445.

[Pentland, 1995]
PENTLAND A. 1995. "Machine Understanding of Human Actions". In *Proceedings of the 7th International Forum: Frontier of Telecommunication Technology.* Nov. 95, Tokyo, Japan. pp. 757-764.

[Peters et al., 2001]
PETERS B.O., PFURTSCHELLER G. AND FLYVBJERG H. 2001. „Automatic Differentiation of Multi-Channel EEG Signals". In *IEEE Transactions on Biomedical Engineering, 48(1)*: 111-116.

[Pfurtscheller, 1977]
PFURTSCHELLER G. 1977. "Graphical Display and Statistical Evaluation of Event-related Desynchronization" In *Electroencephalography and Clinical Neurophysiology, 43*: 757-760.

[Pfurtscheller, 1992]
PFURTSCHELLER G. 1992. "Event-related Synchronization (ERS): an electrophysiological correlate of cortical areas at rest" In *Electroencephalography and Clinical Neurophysiology, 83*: 62-69.

[Pfurtscheller, 1999]
PFURTSCHELLER G. 1999. "EEG Event Related Desynchronization (ERD) and Event Related Synchronization (ERS)". In *Electroencephalography: Basic Principals, Clinical Applications and Related Fields*, 4th Edition Baltimore, Williams and Wilkins. Eds.: Niedermeyer, Lopes da Silva. pp. 958-967.

[Pfurtscheller, 1999a]
PFURTSCHELLER G. 1999. "Quantification of ERD and ERS in the Time Domain. Event-related Desynchronization and Related Oscillatory Phenomena of the Brain" In *Handbook of Electroencephalography and Clinical Neurophysiology, vol. 6, revised Edition*, Eds.: Pfurtscheller, Lopes da Silva. Elsevier, Amsterdam. pp. 89-106.

[Pfurtscheller and Aranibar, 1977]
PFURTSCHELLER G. AND ARANIBAR A. 1977. „Event-related Cortical Desynchronization Detected by Power Managements of the Scalp EEG" In *Electroencephalography and Clinical Neurophysiology, 42*: 817-826.

[Pfurtscheller and Aranibar, 1978]
PFURTSCHELLER G. AND ARANIBAR A. 1978. „Änderungen in der spontanen EEG Aktivität vor Willkürbewegungen. Neue Wege bei der Untersuchung der zentralen µ-Aktivität". In *Zeitschrift für Elektroenziphalographie und Magnetoenziphalographie, 9*: 18-23.

[Pfurtscheller and Lopes da Silva, 1999]
PFURTSCHELLER G., LOPES DA SILVA F.H. 1999. „Event-related EEG/MEG Synchronization and Desynchronization: Basic principles" Invited Review. In *Clinical Neurophysiology 110*: 1842-1857.

[Pfurtscheller et al., 1993]
PFURTSCHELLER G., FLOTZINGER D. AND KALCHER J. 1993. „Brain-Computer Interface: A New Communication Device for Handicapped People". In *Journal on Microcomputer Applications, 16*: 293-299.

[Petche et al., 1992]
PETSCHE H., LACROIX D., LINDNER K., RAPPELSBERGER P. AND SCHMIDT-HEINRICH E. 1992. „Thinking with Images and Thinking with Language: A Pilot EEG Probability Mapping Study". In *International Journal on Psychophysiology, 12*: 31-39.

[Picton, 1992]
PICTON T.W. 1992. “The P300 Wave of the Human Event-Related Potential”. In *Journal on Clinical Neurophysiology, 9*: 456-479.

[Polich, 1999]
POLICH J. 1999. “P300 in Clinical Applications”. In *Electroencephalography: Basic Principals, Clinical Applications, and Related Fields, 4th Edition.* Eds.: E. Niedermeyer, F. Lopes da Silva. Williams and Wilkins, Baltimore. pp. 1073-1091.

[Quek et al., 2002]
QUEK F., MCNEILL D., BRYLL R., DUNCAN S., MA X.F., KIRBAS C., MCCULLOUGH K.E. AND ANSARI R. 2002. “Multimodal Human Discourse: Gesture and Speech”. In *ACM Transactions on Human Computer Interaction, 9(3)*: 171-193.

[Ramachandran et al., 1999]
RAMACHANDRAN V.S., ALTSCHULER E.L., STONE L., AL-ABOUDI M., SCHWARTZ E. AND SIVA N. 1999. “Can Mirrors alleviate visual hemineglect?” In *Medical Hypotheses, 52(4)*: 303-305.

[Ravden and Polich, 1999]
RAVDEN D. AND POLICH J. 1999. “On P300 Measurement Stability, Habituation, Intra-Trial Block Variation and Ultradian Rhythms”. In *Biological Psychology, 51*: 59-76.

[Razzolatti et al., 1996]
RAZZOLATTI G., LUPPINO G. AND MATELLI M. 1996. “The Classic Supplementary Motor Area is Formed by Two Independent Areas”. In *Advances in Neurology, 70*: 45-56.

[Ruchkin et al., 1964]
RUCHKIN S.D., VILLEGAS J AND JOHN E.R. 1964. “An Analysis of Average Evoked Potentials Making Use of Least Mean Square Techniques”. In *Annuals New York Academy of Sciences, 15*: 799-826.

[Salmoni et al., 1984]
SALMONI A.W., SCHMIDT R.A. AND WALTER C.B. 1984. “Knowledge of Results and Motor Learning: A Review and Critical Reappraisal”. In *Psychological Bulletin, 95*: 355-386.

[Sayers et al., 1974]
SAYERS B. MC.A, BEAYLEY H.A. AND HENSHALL W.R. 1974. “The Mechanism of Auditory Evoked EEG Responses” In *Nature 247*: 481-483.

[Schalk et al., 2000]
SCHALK G., WOLPAW J.R., MCFARLAND D.J. AND PFURTSCHELLER G. 2000. “EEG Based Communication: Presence of an Error Potential”. In *Clinical Neurophysiology 111*: 2138-2144.

[Schalk et al., 2004]
SCHALK D.J., MCFARLAND T., HINTERBERGER N., BIRBAUMER N. AND WOLPAW J.R. 2001. „BCI2000: Development of a General Purpose Brain-Computer Interface (BCI) System“. In *Society for Neuroscience: Abstracts, 27*: 63.19, p.168.

[Scheffers et al., 1996]
SCHEFFERS M.K., COLES M.G.H., BERNSTEIN P.S., GEHRING W.J. AND DONCHIN E. 1996. „Event-Related Brain Potentials and Error-Related Processing: An Analysis of Incorrect Responses to Go and No-Go Stimuli“. In *Psychophysiology, 51*: 109-128.

[Sedgewick, 1998]
SEDGEWICK R. 1998. “Algorithms in C/C++”. Eds.: Gordon P.S., Lafferty D., Willcutt A., Dupré L. Addison-Wesley. vol.1,ch.4. pp. 127ff.

[Segal and Fusella, 1970]
SEGAL S.J. AND FUSELLA V. 1970. “Influences of Imagined Pictures and Sounds on Detection of Visual and Auditory Signals”. In *Journal of Experimental Psychology, 83*: 458-464.

[Sharbrough et al., 1991]
SHARBROUGH F., CHATRIAN G.-E., LESSER R.P., LÜDERS H., NUWER M. AND PICTON T.W. 1991. “American Electroencephalographic Society Guidelines for Standard Electrode Position Nomenclature”. In *Journal of Clinical Neurophysiology, 8*: 200-202.

[Shepard and Metzler, 1971]
SHEPARD R.N. AND METZLER J. 1971. “Mental Rotation of Three-Dimensional Objects”. In *Science, 171*: 701-703.

[Singer, 1993]
SINGER W. 1993. “Synchronization of Cortical Activity and its Putative Role in Information Processing and Learning” In *Annual Reviews on Physiology, 55*: 349-374.

[Smith and Smith, 1987]
SMITH T.J. AND SMITH K.U. 1987. “Feedback Control Mechanisms of Human Behavior”. In *Handbook of Human Factors.* Ed.: G. Salvendy. Wiley, New York.

[Spencer et al., 1999]
SPENCER K., DIEN J. AND DONCHIN E. 1999. “A Componential Analysis of the ERP Elicited by Novel Events Using a Dense Electrode Array” Special Report. In *Psychophysiology, 36(3)*: 409-414.

[Squires et al., 1975]
SQUIRES N.K., SQUIRES K.C. AND HILLYARD S. 1975. “Two Varieties of Long-Latency Positive Waves Evoked by Unpredictable Auditory Stimuli in man”. In *Electroencephalography and Clinical Neurophysiology, 41*: 449-459.

[Steinschneider et al., 1994]
STEINSCHNEIDER M., SCHRÖDER C.E., AREZZO J.C. AND VAUGHAN H.G. JR. 1994. „Speech-Evoked Activity in Primary Auditory Cortex: Effects of Voice Onset Time“. In *Electroencephalography and Clinical Neurophysiology, 92*: 30-43.

[Steriadis and Constantinou, 2003]
STERIADIS C.E. AND CONSTANTINOU P. 2003. “Designing Human-Computer Interfaces for Quadriplegic People”. In *ACM Transactions on Computer-Human Interaction, 10(2)*: 87-118.

[Streeter and Raviv, 1965]
STREETER D.N. AND RAVIV J. 1965. “Research on Advanced Computer Methods for Biological Data Processing”. Rept ASTIA, Doc AD 637452. Springfield, VA: Defence Documentation Center, Defence Supply Agency, Clearing House of Federal Scientific and Technical Information.

[Sutter, 1992]
SUTTER E.E. 1992. “The Brain Response Interface: Communication through visually induced Electrical Brain Responses”. In *Journal of Microcomputer Applications, 15*: 31-45.

[Sutton et al., 1965]
SUTTON S., BRAREN M., ZUBIN J. AND JOHN E.R. 1965. “Evoked Potentials Correlates of Stimulus Uncertainty”. In *Science 150*: 1187-1188.

[Tanji, 1994]
TANJI J. 1994. “The Supplementary Motor Area in the Cerebral Cortex”. In *Neuroscience Research*, 19: 251-268.

[Vapnik, 1995]
VAPNIK V.N. 1995. “Statistical Learning Theory” Edt.: Haykin S. Springer, NY.

[Vidal, 1973]
VIDAL J.J. 1973. “Towards Direct Brain-Computer Communication”. In *Annual Review of Biophysics and Bioengineering*. pp. 157-180.

[Vijn et al., 1991]
VIJN P.C.M., VAN DIJK B.W. AND SPEKREIJSE H. 1991. “Visual Stimulation reduced EEG activity in Man” In *Brain Research, 550*: 49-53.

[Voelker et al., 1989]
VOELKER S.L., CARTER R.A., SPRAGUE D.J, GDOWSKI C.L. AND LACHAR D. 1989. “Developmental Trends in Memory and Meta-Memory in Children with Attention Deficit Disorders”. In *Journal of Pediatric Psychology, 14*: 75-88.

[Waterhouse and Robinson, 1994]
WATERHOUSE S.R. AND ROBINSON A.J. 1994. “Classification Using Hierarchical Mixtures of Experts”. In *Procedings of the IEEE Workshop on Neural Networks for Signal Processing*. pp. 177-186.

[Wolpaw et al., 1991]
WOLPAW J.R., MCFARLAND D.J., NEAT G.W. AND FORNERIS C.A. 1991. “An EEG-Based Brain-Computer Interface for Cursor Control”. In *Electroencephalography and Clinical Neurophysiology, 78*: 252-259.

[Wolpaw et al., 2002]
WOLPAW J.R., BIRBAUMER N., MCFARLAND D.J., PFURTSCHELLER G. AND VAUGHAN T.M. 2002. “Brain-Computer Interfaces for Communication and Control”. In *Clinical Neurophysiology, 113*: 767-791.

[Черёмушкин et al., 2001]
ЧЕРЁМУШКИН А.В., АЛФЕРОВ А.П., КУЗЬМИН А.С. AND ЗУБОВ А.Ю. 2001. «Основы Криптографии. Учебное пособие». Helios APB, Moscow, Russia.

GLOSSARY

A/D	—	Analog/Digital
ADT	—	Abstract Data Type
ALS	—	Amyotrophic Lateral Sclerosis
BBCI	—	Berlin Brain-Computer Interface
BCI	—	Brain-Computer Interface
BP	—	Bereitschaftspotential (see also: LRP)
bpm	—	Bits per Minute
BRI	—	Brain-Response Interface
CAT / CRT	—	Command Activation / Relaxation Term
CATh	—	Command Activation Threshold
CHI / HCI	—	Computer-Human / Human-Computer Interaction
CNS	—	Central Nervous System
CSI	—	Client-Server-Interface
DOF	—	Degree of Freedom
DRS	—	Distributed Rendering System.
DSP	—	Digital Signal Processing
EEG	—	Electroencephalography
EMG / EOG	—	Electromyogram / Electrooculogram
ERD / ERS	—	Event-Related Desynchronization / Synchronization
ERP	—	Event-Related Potentials
FFT / FT	—	(Fast) Fourier Transformation

fMRI	—	functional Magnetic Resonance Imaging
FP / FN	—	False Positive / False Negative
GUI	—	Graphical User Interface
HCI	—	Human-Computer Interaction
ISM	—	Internal Simulation of Movements
LAN/WAN	—	Local Area Network / Wide Area Network
LDA / QDA	—	Linear / Quadratic Discriminant Analysis
LMC	—	Lateral (pre-)Motor Cortex
LPM	—	Linear Programming Machine
LRP	—	Lateralized Readiness Potential (see also: BP)
MI	—	Primary Motor Cortex
MEG	—	Magnetoencephalography
MFC	—	Microsoft Foundation Classes
N100 / P300	—	Negativation / Positivation at about 100 ms / 300 ms after stimulus, respectively
OOD / OOP	—	Object Oriented Design / Programming
PET	—	Positron Emission Tomography
RFD / FD	—	(Regularized) Fisher Discriminant
RDA	—	Remote Data Access
RGB	—	Red-Green-Blue; a common color coding standard
SCP	—	Slow Cortical Potential
SNR	—	Signal-to-Noise Ratio
SMA	—	Supplementary Motor Area
SVM	—	Support Vector Machine
TCP/IP	—	Transmission Control Protocol / Internet Protocol
TTD	—	Thought Translation Device
UDP	—	User Datagram Protocol
UML	—	Unified Modeling Language
VEP	—	Visual Evoked Potential
VR	—	Virtual Reality
WTA	—	Winner Takes All