Masterarbeit
am **Institut für Bauinformatik**

TU Graz
Graz University of Technology

Evaluierung der Möglichkeiten von kostengünstigen Brain-Computer-Interfaces

bezüglich der Eignung
zur freihändigen Bedienung von Computern

vorgelegt von | **Martin Krammer**, BSc
Studienrichtung | Wirtschaftsingenieurwesen – Bauingenieurwissenschaften

Betreuer: Univ.-Prof. Dr.techn. Dipl.-Bauing. Ulrich Walder;
Dipl.-Ing. Rüdiger Schütz; Thomas Bernoulli, MSc

November 2009

Layout: Martin Krammer
Titelbild: `http://www.brainfingers.com/images/leonard.jpg` (Stand Sept. 2009)

Kurzfassung

Titel: Evaluierung der Möglichkeiten von kostengünstigen Brain-Computer-Interfaces
– bezüglich der Eignung zur freihändigen Bedienung von Computern

Tätigkeitsauslöser Gehirn

Das menschliche Gehirn ist der Ausgangspunkt willentlicher Körperaktionen. Die Übermittlung der Steuersignale in die betreffenden Körperregionen geschieht danach über das weit verzweigte Nervensystem. Diese menschliche Informationsverarbeitung und -übertragung basiert hauptsächlich auf der Weitergabe von elektrischen Potentialen. Eine dadurch ausgelöste Körperbewegung kann letztendlich z. B. der Benutzung eines Gerätes dienen.

Gehirn – Maschine Interaktion

Brain-Computer-Interfaces (BCI) versuchen den motorischen Part des Handlungsablaufes zu minimieren oder überhaupt zu beseitigen. Dadurch bleiben sonstige physische Tätigkeiten von der Gerätesteuerung unbeeinflusst. Jede manuell geprägte Arbeit, welche darüber hinaus von Computersteuerungsaufgaben begleitet ist, kann vom Einsatz eines BCI profitieren, da der eigentliche Arbeitsablauf ungestört(er) bleibt.

charakteristische Gehirntätigkeit

Solche Geräte nutzen die elektrischen Schwankungen infolge Gehirnaktivität. Die verschiedenen Areale des Gehirns sind jeweils für definierte Körperfunktionen zuständig. Hirnsignale zeigen sich situationsabhängig (Schlaf, Aufregung, Alter etc.) charakterisisch bezüglich Frequenz, Amplituden, Muster etc. Die, aus der Medizin stammende, Kenntnis der Lokalität bzw. der Charakteristik ermöglicht die von BCI angestrebte Umsetzung einer „direkt(er)en“ Computersteuerung. BCI sind noch Teil wissenschaftlicher Forschung und es existieren verschiedenste Lösungsansätze.

Hirnströme – EEG-Messung

Ein Großteil der Systeme stützt sich auf die Hirnstrommessung über Elektroenzephalografie (EEG), deren Frequenzen von Computern interpretiert/klassifiziert wird. Andere Potentialquellen, die eine EEG-Messung beeinträchtigen können (z. B. durch Muskel- bzw. Augenbewegungen, Herzschlag, Stromnetz etc.) werden von diesen Systemen als Störung behandelt („Artefakte“). Die verwendeten EEG-Messeinrichtungen stammen aus der langjährigen medizinischen Forschung und sind dadurch sehr ausgereift, jedoch auch dementsprechend kostspielig.

EEG-BCI: schwieriges Erlernen

Die Stützung der Systeme auf reine Hirnstromcharakteristika birgt jedoch auch Probleme hinsichtlich der Anwenderfreundlichkeit. Es zeigt sich, dass der Benutzer eine oft schwierige Lernphase zu überwinden hat, um seine Hirnsignale – dem Brain-Computer-Interface entsprechend – konsistent einzusetzen. Die Steuerung interner physiologischer Vorgänge ist nicht alltäglich und verlangt nach Übung. Auch scheinen

viele Personen physiologisch gar nicht dafür geeignet zu sein (Krankheiten) bzw. benötigen noch mehr Übung als „Talentierte“.

kostengünstiger Ansatz

Der in dieser Arbeit vorgestellte Ansatz beinhaltet als Hauptpunkt die Verwendung von kostengünstiger Hardware. Gelänge damit in gewissen Bereichen eine zuverlässige Umsetzung der genannten BCI-Ziele („freihändig“), so steht einem sofortigen Einsatz in passenden Anwendungen nichts im Wege. Zumindest lohne sich dann oftmals auch die Inbetrachtziehung von BCI als Ersatz oder Unterstützung herkömmlicher Eingabesysteme.

Evaluierung OCZ NIA

Ein Gerät, welches BCI-Eigenschaften verspricht und preislich attraktiv ist, wurde Anfang 2008 von der Firma OCZ Technology Inc. auf den Markt gebracht. Es handelt sich um ein Computereingabegerät zum unterstützenden Einsatz in Computerspielen. Ein kompletter Maus- bzw. Tastaturersatz soll der NIA allerdings nicht sein. Der NIA verwendet als Steuersignal nicht nur Hirnstromdaten (EEG); in Wirklichkeit gründet seine Funktionsweise auf einer Mischung von Muskel-, Augen- und Hirnpotentialen – kurz Biopotentialen.

ungeeignete Original-Software

Die Umsetzung der originalen NIA-Steuersoftware bzw. insbesondere das zugrundeliegende (Spiele-)Konzept, zeigte sich für die in dieser Masterarbeit untersuchte, leicht erlernbare, allgemeine, freihändige Computersteuerung leider als weitgehend ungeeignet. Im Zuge der Arbeiten mit dem NIA wurde allerdings erkannt, dass Augen- bzw. Stirnmuskel-Artefakte charakteristische Muster in den NIA-Ausgabedaten erzeugten. Diese Beobachtung war der Ausgangspunkt zur Entwicklung einer eigenen, unabhängigen Steuersoftware für den NIA.

eigene Softwareentwicklung

Software lernt vom Benutzer

Im Vergleich zum EEG-BCI-Ansatz, erscheinen dabei die leichte Erlernbarkeit und Nachvollziehbarkeit dieser natürlichen Körperbewegungen vorteilhaft zu sein. Nachteilig ist, dass die Zahl jener Bewegungen, welche charakteristische Muster auslösen, beschränkt ist. Weiters ist die Ausprägung der Muster sehr benutzerabhängig. Die Software verwendet Methoden der digitalen Signalverarbeitung und Computational Intelligence, um diese Muster bestimmten Befehlen zuordnen zu können. Dass sich die Software dabei auf das Verhalten des jeweiligen Benutzers anpasst (nicht umgekehrt), war der entscheidende Verbesserungsvorschlag im Gegensatz zur originalen NIA-Software. Die Vorgehensweise kann als ausbaufähiges Konzept für Applikationen gesehen werden, wenn eine freihändige Computerbedienung kostengünstig umgesetzt werden soll.

Diese Masterarbeit beschreibt also neben dem BCI-Prinzip sowie verwendeten Systemen, hauptsächlich den NIA, die vom Autor entwickelte NIA-Software bzw. welche Vorteile aber auch Widrigkeiten bestehen.

Abstract

Title: Evaluation of Low-Cost Brain-Computer-Interfaces
– as a Means for General-Purpose Hands-Free Computer-Control

The human brain is the origin of voluntary body-actions. The transmission of generated control signals to the relevant areas of the body happens via the nervous system. This information-processing and -transmission is mainly based on electric potentials. Body movements induced by such a „chain of command“ may ultimately serve the usage of a device.

brain-activity triggers body-actions

The purpose of a Brain-Computer-Interface (BCI) is to minimize the motor part of the described body-action sequence or eliminate it completely. Other physical activities will therefore remain relatively independent of device-controlling-actions. If a user is carrying out primary tasks that require his undisturbed physical attention, a BCI-device can be used to support a secondary task of interacting with a computer.

Brain – Computer Interaction

BCI devices function based on the characteristics of brain-activity. Medical research has revealed that the location and electrical properties of brain signals are indicative for the intended task. The „functionality“ in the human brain is located in several areas. Each area is responsible for different body-related tasks. The electrical properties of the brain-activity are related to the specific situation wherein a task is carried out (sleep, excitement, age etc.). The characteristics and locality of brain-signals are well known because of medical research. These findings like frequency- or amplitude-patterns are basically the foundations for the wanted direct link from the brain to the computer.

characteristic brain-activity

Most BCI-systems try to use the electrical brain-activity-characteristics measured by electroencephalography (EEG). Other electrical signals which could be registered (such as muscle-activity, eye-movement, heart-rate or power-supply system) are generally treated as disturbances („artefacts“). EEG-systems are highly accurate and expensive medical devices.

brain-activity – EEG

It is important to realize that using (only) EEG data for BCI has its useablility-drawbacks. Brain-activities are „internal“ physiological events and people are not used to or normally not supposed to control them at will. Users have to learn a completely new way of doing something and that can take a long time (shallow learning curve). There are people who are physiologically better suited for this kind of internal control and others who won't be able to achieve the goal at all (diseases).

EEG-BCI: useability-problems

One of the key-points of the approach described within this master-thesis is the use

low-cost devices

of low-cost hardware. If it is possible to reach the goals of BCI without expensive hardware, it can be adapted for countless mass-market uses. Even if such a system fulfils only a limited set of BCI-requirements, it may be desirable as a supportive hands-free input-device.

OCZ NIA

A BCI-device which became available in early 2008 is the Neural Impulse Actuator (NIA) from OCZ Technology Inc. It is an input-system to support/improve the computer game playing experience. It is not meant to be a complete mouse/keyboard-replacement, but to support the human-computer-interaction in an innovative way. This handy configuration seems to be quite suitable for mobile uses, and therefore the NIA can be seriously considered for additional compelling and general usage-scenarios.

The NIA is not based solely on EEG-data recognition; it is designed to also include muscle and eye-movement-data in its BCI-classification. So the NIA-BCI uses a mixture of these bio-potentials with a high impact coming from artefacts.

During the work, it emerged that the original (game-based) NIA-control-software was not suitable for an easy to learn general-purpose handsfree computer-control as was intended by this thesis.

custom NIA-software

It was observed, however, that frontalis-muscle (forehead) movement and eye movement showed distinctive patterns within the voltage output of the NIA. This behaviour generated the idea of a custom control-software development for the NIA. This software uses common algorithms for digital-signal-processing and computational-intelligence to map the patterns to defined commands.

user-adaptive concept

It was discovered that the patterns are highly user-dependent. Therefore it was necessary to implement the software so that the user's individual skills and generated patterns would be „learned“ by the computer. This user-adaptive property of the software is one of the most important distinctions to the original NIA-software. This approach has to be seen as an extensible concept for applications where an easy-to-learn low-cost handsfree computer-control is needed.

This master-thesis introduces the basic principles and current systems, used for Brain-Computer-Interfaces. The main part includes a thorough evaluation of the low-cost gaming-BCI NIA as a means to control general computer-software other than games. A therefore developed custom control-software will be described and discussed as much as problems and other findings.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

1 Einleitung

In der vorliegenden Arbeit wird ein Überblick bezüglich *Brain-Computer-Interfaces (BCI)* geboten. BCI sind im Wesentlichen Systeme, welche eine *freihändige Bedienung* von Geräten mithilfe des *Computers* ermöglichen. Die bewusste Steuerung basiert üblicherweise auf einer direkten Messung und Interpretation an der Quelle der Willensbildung – dem *menschlichen Gehirn*.

nicht-manuelle Steuerung

1.1 Motivation

Die Hintergründe dieser Arbeit liegen in der Idee, in *Menüsystemen* von Softwareanwendungen mittels BCI navigieren zu können. Der Reiz der Paarung von *Innovation*, *Erleichterung* und *Leistbarkeit* weckt mannigfaltiges Anwendungsinteresse für BCI-Steuerungen. Deshalb erhalten in dieser Arbeit auch die kürzlich am Markt erschienenen, *kostengünsten BCI-Geräte* aus der Computerspiele-Industrie besondere Aufmerksamkeit. Die Arbeit dient als Einblick in BCI und soll Möglichkeiten für die Forschungstätigkeit am *Institut für Bauinformatik* der *Technischen Universität Graz* aufzeigen.

Menüsysteme

Low Cost-BCI

Die Notwendigkeit der Interaktion mit Computern im Zuge *manuell bestimmter Arbeiten* oder *Reaktionsschnelligkeit* fordernder Tätigkeiten sind potentielle Einsatzgründe für BCI-Systeme. Zudem kann die Computerbedienung in manchen Fällen (z. B. *körperlicher Dysfunktionen*) überhaupt erst durch BCI ermöglicht werden. Eine mögliche Verknüpfung dieser Punkte mit *hohen Personenzahlen* erklärt die darüberhinausgehende Relevanz von niedrigen Gerätestückkosten. Beispielhaft können damit folgende Anwendungen identifiziert werden:

vielfältiger Einsatz

- *Bauarbeiter*, welche während der Arbeit in elektronischen Planunterlagen navigieren.
- *Rettungskräfte*, welche während ihres Einsatzes zur Orientierung an potentiell unbekannten, obskuren Orten in elektronischen Karten bzw. Plänen Informationen finden.

manuell belegte Tätigkeiten

- *Behinderte Menschen*, denen eine Interaktion mit dem Computer ohne BCI überhaupt nicht möglich wäre. — körperliche Beeinträchtigung
- *Militärische Anwendung* zur oft komplexen Steuerungsunterstützung in Kampfgeräten. Der Einsatz von BCI als Bedienmöglichkeit bei (vielen) elektronisch aufgerüsteten Soldaten verlangt zudem nach kostengünstigen Geräten. — Mannstärke
- *Computerspieler*, welche zumeist vielseitige Tätigkeiten gleichzeitig durchführen müssen. Weiters sind dort rapide Reaktionen für den Spielerfolg essentiell. — Reaktionsschnelligkeit

Allemal zielt der Einsatz von BCI auf die *Entlastung* körperlich ausgeführter Steuerungsaufgaben ab. Je nach Anwendungsfall und Benutzerpräferenz rücken wechselnde Prioritäten in den Fokus. Manche definieren eine hohe *Zuverlässigkeit* weniger Funktionen als unumgänglich, andere wollen *schnelle und vielseitige Steuerung*, können dagegen jedoch zeitweise *Fehlfunktionen* tolerieren. — variable Ansprüche an BCI

Auch zeitliche Aspekte können nur differenziert beantwortet werden; wie lange ein *Erlernen der BCI-Funktionen* dauern darf bzw. wie *schnell* das System (zuverlässig) *einsatzfähig* sein muss, hängt von der Verwendung ab. — Zeitaufwand

1.2 Ziele der Arbeit

Vorrangig sollen die Grundlagen von BCI-Anwendungen vermittelt werden. Dabei spielen verschiedenste Wissensgebiete wie z. B. die *Medizin*, *Mathematik*, *Informatik* und *Elektrotechnik* sowie *Ergonomie* eng vernetzte Rollen. Es müssen Fragen nach — Grundverständnis

- *Entstehung* von Steuersignalen im Körper („Biosignalquelle“),
- deren *Registrierung* und *Leitung* durch elektrische Bauteile,
- der Behandlung in der *Elektronik*,
- einer anschließenden *Aufbereitung* und *Interpretation* durch *Software* sowie
- der Bedienung über Steuerungssoftware (*Useability*)

Interdependenzen

geklärt werden. Letztlich soll die Ausführung der gewünschten Tätigkeit korrekt induziert werden.

Darüber hinaus werden die Probleme von Störsignalen geklärt, jedoch auch Möglichkeiten zur Nutzung und Kombination weiterer Biosignale aufgezeigt. Die Messung dieser Signale, welche Interpretationsmöglichkeiten sie bieten und wo Schwierigkeiten liegen, wird im Laufe der Arbeit bewusst gemacht.

Generell soll besprochen werden, was über BCI möglich ist bzw. was derzeit Ver- — Möglichkeiten

wendung findet. Deshalb wird zum Stand der Wissenschaft Stellung genommen, verwendete Methoden aufgezeigt sowie die Marktsituation analysiert.

OCZ NIA

Nach Erarbeitung der theoretischen Grundlagen lag besonderes Augenmerk auf einem kostengünstigen BCI-Gerät aus der Spielebranche; dies war der *NIA – Neural Impulse-Actuator* von *OCZ Technology Inc.* Er stand im Zuge der Arbeit zur Verfügung und war Gegenstand von praktischen Experimenten. Die Versuche umfassten zu Beginn eine Anwendungserprobung mit der, dem Gerät beigelegten, *Original-Software*. Aufgrund des Wunsches nach einer leicht erlernbaren BCI-Menüsystem-Steuerungsmöglichkeit musste die Spiele(r)-orientierte NIA-Software dahingehend jedoch als *ungeeignet eingestuft* werden.

Erprobung

Software-entwicklung

Dies bekräftigte den Entschluss zur Entwicklung einer *eigenen* NIA-Computeranwendung namens *NiaBlinkReader*. Der NiaBlinkReader liest die NIA-Biosignaldaten ohne zusätzliche Treibersoftware aus. Methoden der *Computational Intelligence* schaffen daraufhin die Interpretation/*Klassifikation* von Biosignalen durch *Lernen vom Benutzer*. Die Software entstand aus Überlegungen zur vorangegangenen Theoriediskussion, dient der Findung von passenden Biosignalverarbeitungsmethoden und weist deshalb *experimentelle Züge* auf.

Forschung

Das Programm kann als Grundlage für *weitere Forschung* in Richtung „BCI zur Steuerung einfacher Menüsysteme“ dienen. Bei Betrachtung der Erkenntnisse können damit gewisse Fragestellungen vorweggenommen bzw. eine „Entwicklungsrichtung“ eingeschlagen werden.

1.3 Aufbau der Arbeit

umfassende Beschreibung

Resultierend aus der Unterschiedlichkeit der beteiligten Wissengebiete entstehen häufig Querverbindungen, welche ohne vielseitige Beschäftigung mit dem Thema nicht einfach einzusehen sind. Um die logischen Schlüsse aus den Abhängigkeiten vermitteln zu können, mussten daher alle, die BCI betreffenden, Wissensgebiete einen adäquaten Eingang in die Arbeit finden.

Theorie

Folglich führt die theoretische Aufarbeitung von BCI in Abschnitt 3 erst über die *menschliche Physiologie* in Abschnitt 2. Darin werden die relevanten Dinge zum *Aufbau* und der *Informationsverarbeitung* des *Gehirns* ebenso abgehandelt, wie die, in der *Medizintechnik* eingesetzten, Instrumente zur Hirnsignalmessung sowie deren Funktionsweise. Diese beiden Abschnitte sind Auszüge aus dem vorangegangenen Masterprojekt des Autors *„Zu den Grundlagen und Möglichkeiten*

von Brain-Computer-Interfaces – Stand der Technik und Einsatzszenarien aktueller Geräte in der freihändigen Bedienung von Computern“ und dienen der Einführung in das Thema BCI bzw. dem Verständnis zur folgenden Evaluierung des NIAs.

In den Abschnitten 4 und 5 werden dann die *Hardware* bzw. originale/selbst erstellte *Software* des NIAs eingehend vorgestellt. Aus durchgeführten *Benutzertests* konnten Probleme in der Verwendung des NIAs herausgearbeitet werden. Diese und weitere *Erkenntnisse* sind in Abschnitt 6, gegliedert nach Zugehörigkeit, vermerkt. Den Abschluss bildet Abschnitt 7, wo eine kurze *Zusammenfassung und Empfehlungen* gegeben werden.

Praxis
Tests
Fazit

2 Allgemeines zur Hirnsignalmessung

Das Gehirn ist die „Steuerzentrale“ des Menschen. Aus Umwelt- oder *Körper-Informationen* entstehen nach der Verarbeitung im Gehirn *Körper-Steuersignale*. Im Sinne eines *Regelkreises* ist das Gehirn der *Regler*, welcher die *Regelgrößen* über verschiedenste *Stellgrößen* in den Soll-Zustand bringt.

Informationen
⇓
Steuersignale

Zum Informationsaustausch zwischen den verschiedenen Körperregionen besitzt der Mensch das *Nervensystem*. Dieses wird in das *Zentralnervensystem* (Gehirn, Rückenmark) bzw. das *periphere Nervensystem* unterteilt und besteht aus der Verbindung vieler Nervenzellen, sog. *Neuronen*. Die Signalübertragung basiert auf *elektrochemischen Vorgängen*.

Verteilung

Elektrizität

System-Informationen gelangen also über das Nervensystem in das Gehirn. Aus der dortigen Verarbeitung zeigen sich oftmals notwendige Korrekturmaßnahmen, weshalb vom Gehirn entsprechende Werte für die Stell-/Regelgrößen definiert werden. Das Nervensystem leitet diese Werte zu den entsprechenden Körperregionen; dies können z. B. Muskeln für *Bewegungen* oder *Organfunktionen* sein, *Hormondrüsen* etc. Durch die Reaktionen entsteht ein Steuersystem, welches über *Rückkopplung* funktioniert.

Rückkopplung

2.1 Gehirnphysiologie

Im menschlichen Gehirn befinden sich ca. $10^{10}-10^{11}$ über *Synapsen* eng vernetzte Neuronen; sie sind für die Informationsverarbeitung verantwortlich. Das Gehirn wiegt etwa 1500 g und ist in verschiedene Bereiche (siehe Abbildung 2.1, links) unterteilt.[1]

$\sim 10^{10}$-10^{11} Neuronen

$\sim$1.5 kg

Das *Cerebrum* ist der evolutionär jüngste und größte Teil des Gehirns. Es besteht aus zwei Hälften (*Hemisphären*), wobei die rechte Seite Informationen der linken Körperseite verarbeitet und umgekehrt.[2]

[1] siehe [Thompson 2001]
[2] siehe [Nickolls 2002, S. 1]

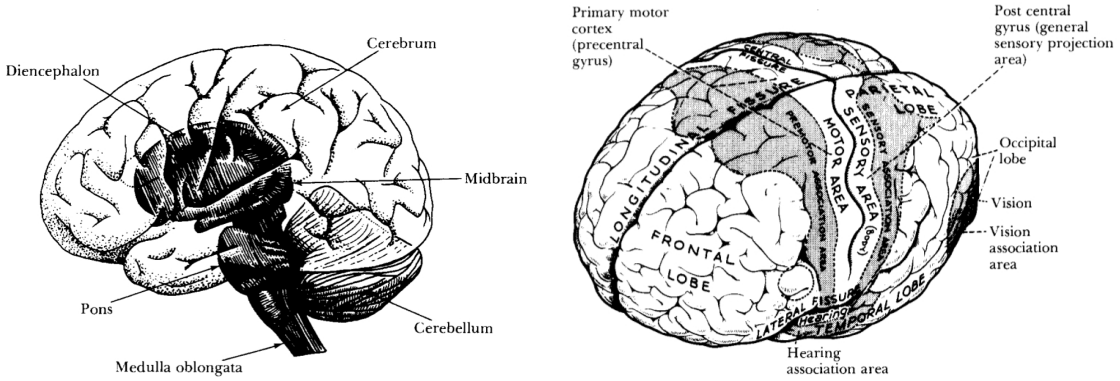

Abbildung 2.1: Die Anatomie des Gehirns [Nickolls 2002, S. 2] (links), Cerebrale Hemisphären [Nickolls 2002, S. 3] (rechts)

In den äußersten Schichten (ca. 4 mm, *Cortex* oder *Großhirnrinde*) des Cerebrums, also im *Cerebralen Cortex*, sind wichtige Teile des Gedächtnisses, der Aufmerksamkeit, der Sprache oder des Bewusstseins angesiedelt; je größer das verwendete Areal, desto besser ist die jeweilige Funktion ausgebildet (siehe Abbildung 2.2).[3]

Sensorischer / Motorischer Cortex

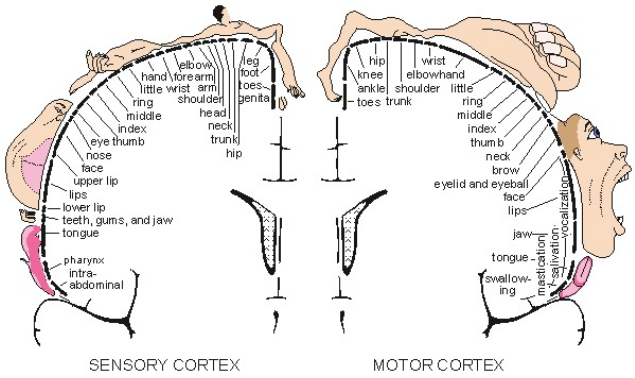

Abbildung 2.2: Funktionen der Areale im *Sensorischen* bzw. *Motorischen Cortex* [Malmivuo und Plonsey 1995, Abb. 5.6]

Die Funktionsweise des Gehirns gründet auf dem Zusammenspiel der vielen Neuronen; die Interaktion geschieht durch Übertragung von *elektrischen Potentialen* zwischen den Neuronen. Die Summe der an einem Neuron eingehenden Potentiale müssen einen gewissen Schwellwert erreichen, um das *Aktionspotential* des betreffenden Neurons auszulösen und eine Weitergabe an nachfolgende Neuronen zu ermöglichen. Je größer der Abstand zwischen Neuronen, desto stärker wird dabei das weitergeleitete Signal abgeschwächt.[4]

Aktionspotential

[3] siehe [Malmivuo und Plonsey 1995, Abs. 5.4.2]
[4] siehe [Malmivuo und Plonsey 1995, Abs. 5.4.3] bzw. [Malmivuo und Plonsey 1995, Abs. 2]

Die komplexe Vernetzung im Gehirn ermöglicht mit dem Mechanismus der Potentialweitergabe die Verarbeitung von Informationen. Bei Bedarf passt sich das Gehirn an und bildet neue Verbindungen; dieses Verhalten spiegelt die bekannte Lernfähigkeit des Menschen wider. Andererseits können jedoch wenig benutzte Synapsen wieder degenerieren („Verlernen“).

vernetzen⇝ erlernen

2.2 Messmethoden

Die Neuronenaktivität ist für viele Situationen charakteristisch und kann gemessen werden. Die Messung der Hirnströme ist ein Interessengebiet der *Neurologie* und bildet das Hauptinstrument zur Diagnose von Hirnkrankheiten wie Epilepsie.[5] Zu den etablierten, *nicht-invasiven neurophysiologischen Untersuchungsmethoden* zählen z. B. *Elektroenzephalografie (EEG)*, *Magnetoenzephalografie (MEG)* sowie *funktionelle Magnetresonanztomografie (fMRT)*.

nicht-invasiv: EEG, MEG, fMRT

EEG und MEG erlauben einen Einblick in das Gehirn, indem sie die (elektromagnetischen) Signale der Neuronen messen. EEG zeichnet die elektrischen Potentialschwankungen im Gehirn über Elektroden *am*[6] Kopf auf. MEG misst dafür das magnetische Feld, welches durch den Fluss des elektrischen Stromes zwischen den Neuronen entsteht. Eine MEG-Messung erfolgt mit Messinstrumenten nahe der Kopfoberfläche und erlaubt eine bessere örtliche Bestimmung der Aktivität (*spatiale Auflösung*) als EEG, bei gleichzeitigem Erhalt der hohen *temporalen Auflösung*. Die MEG-Sensoren sind sehr empfindlich, weshalb die Anwendung zur Störungsvermeidung in einem magnetisch abgeschirmten Raum durchgeführt wird.[7]

Einen anderer Zugang zur Messung der Hirnaktivität beschreitet die fMRT. Eine erhöhte Stoffwechseltätigkeit von aktivierten Hirnregionen bewirkt nämlich einen lokal erhöhten Sauerstoffbedarf, dessen Deckung durch einen gesteigerten Blutfluss erfolgt. Die Neuronenaktivität wird nicht direkt gemessen, sondern vielmehr kann der zugehörige Blutfluss – über die magnetischen Eigenschaften des eisenhaltigen Blutsauerstoffträgers *Hämoglobin* – im Magnetresonanztomografen (siehe Abbildung 2.3, links[8]) verfolgt werden.[9]

fMRT hat physiologisch bedingt nur eine geringe temporale Auflösung, da der erhöhte Blutfluss (*Hyperperfusion*) oft erst bis zu einer Sekunde nach der Neu-

[5] siehe [Nickolls 2002, S. 15], bzw. [Subasi 2006]
[6] Elektrodenplatzierung *im* Kopf ist möglich; dies zählt dann jedoch zu den *invasiven Methoden.*
[7] siehe [Malmivuo und Plonsey 1995, Abs. 14]
[8] Bild: `http://en.wikipedia.org/wiki/File:Varian4T.jpg` (Stand Sept. 2009)
[9] siehe [Schad 2002, S. 659]

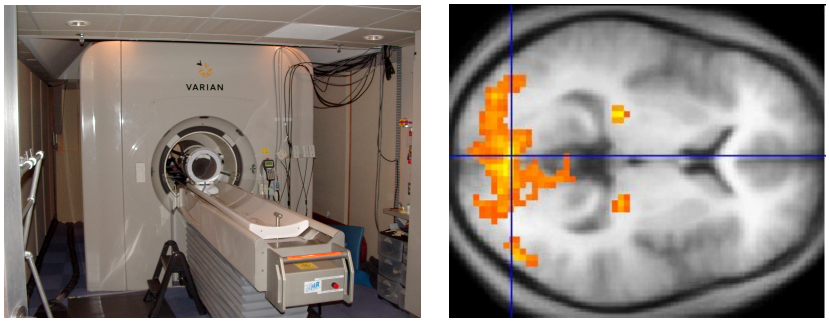

Abbildung 2.3: Magnetresonanztomograf (links),
fMRT-Aufnahme der Gehirnaktivität (rechts)

ronenaktivität auftritt; fMRT bietet dagegen eine sehr hohe spatiale Auflösung (Millimeter-Bereich; siehe Abbildung 2.3, rechts[10]). Im Vergleich reagieren EEG und EMG schneller (Millisekunden-Bereich), zeigen jedoch die Lokalität der betroffenen Hirnareale schlechter (z. B. wegen der Elektrodenabstände).[11]

EEG: flexible Anwendung

EEG ist unter diesen Verfahren sicherlich das kostengünstigste und am einfachsten durchzuführende[12]. Im Verlauf dieser Arbeit wird später öfters auf EEG Bezug genommen, weshalb diese Thematik nachfolgend noch näher erläutert wird.

Nicht-invasive Elektroenzephalografie – EEG

EEG-Daten: $<100\,\mu$Volt; $\sim 1-50$ Hz

Im Falle von EEG werden die Potentialschwankungen der Neuronen durch Elektroden an der Kopfoberfläche registriert (siehe Abbildung 2.4[13]). Die gemessenen Potential-Amplituden reichen bis zu $100\,\mu$Volt und es ergeben sich Frequenzen von ca. $1-50$ Hz[14 15]. Es bedarf dabei ca. 108 Neuronen auf einer Fläche von $6\,\text{cm}^2$ auf dem *Cerebralen Cortex*, um das betreffende EEG-Signal feststellen zu können[16].

Die Anordnung der Elektroden folgt zumeist dem sog. *Internationalen 10-20 System* für 21 Elektroden. Modifizierte Systeme wie das der *American EEG Society* erlauben 75 Elektroden. Jede Variation der Elektroden verändert die Aufzeichnungen, womit die brauchbare Verwendung/Interpretation dem Experten vorbehalten ist. Hoppe vermerkt dazu in [Ebner und Deuschl 2006, S. 18], dass unübliche Konfigurationen

[10] Bild: `http://en.wikipedia.org/wiki/File:FMRI.jpg` (Stand Sept. 2009)
[11] siehe [Darvas et al. 2004, S. 289ff]
[12] siehe [Benimeli und Sharman 2007, S. 361]
[13] Bild links: `http://upload.wikimedia.org/wikipedia/commons/5/5e/EEG_mit_32_Electroden.jpg` (Stand Sept. 2009)
Bild rechts: `http://en.wikipedia.org/wiki/File:EEG_cap.jpg` (Stand Sept. 2009)
[14] siehe [Malmivuo und Plonsey 1995, Abs. 13.1]
[15] Nickolls spricht in [Nickolls 2002, S. 4] dagegen von $1-100$ Hz.
[16] siehe [Olejniczak 2006, S. 187]

EEG-Daten oft eher undurchsichtiger machen, als daraus vorteilhaftere Erkenntnisse gezogen werden können.[17]

Abbildung 2.4: unterschiedliche EEG-Elektrodenanordnungen

Forschungsprojekte verwenden bis zu 250 Elektroden. Es ist verständlich, dass die Anzahl der Elektroden die spatiale Auflösung der Potentialaufzeichnung bestimmt (engere Anordnung gibt höhere Auflösung, aber ebenso vermehrte Beeinflussung zwischen den Elektroden [„rauschen"])[18]. Verdichtete Anordnungen werfen zudem Fragen der Elektrodenbefestigung bzw. -verbindung auf (*Konfiguration*). Die Messung kann prinzipiell auf zwei Arten erfolgen: Einerseits existiert die *referenzielle Erfassung* der EEG-Kanäle („unipolar"), andererseits die *bipolare Methode*. Die referenzielle Konfiguration erfordert eine Referenzelektrode an einem elektrisch relativ „ruhigen" Ort, welcher aber schwierig zu finden ist (daher oft Bildung des Durchschnittswertes mehrerer Referenzelektroden). Bipolare Anordnungen verbinden immer zwei Elektroden und verbessern die örtliche Unterscheidungsmöglichkeit von Hirnaktivitäten, können jedoch Verzerrungen in der Aufzeichnung bewirken (siehe Abbildung 2.5).[19]

referenzielle Elektrodenanordnung

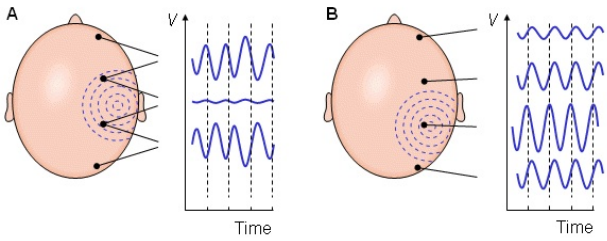

Abbildung 2.5: Unterschiede durch (A) bipolare und (B) unipolare EEG-Aufzeichnung. [Malmivuo und Plonsey 1995, Abb. 13.3]

Die aufgezeichneten EEG-Daten werden in einem *Elektroenzephalogramm* (siehe Abbildung 2.6) dargestellt und können von Experten auf Anomalien (bezüglich *Wellenform, Wiederholungen, Frequenz, Amplitude, Verteilung*)[19] untersucht werden. Solche Anomalien können Anzeichen für z. B. *Hirnschäden, Tumore, Schlaganfälle, Multiple Sklerose* oder *Epilepsie* sein.[20]

EEG: Krankheitsindikator

[17] siehe [Niedermeyer und da Silva 2005, S. 140ff]
[18] siehe [Malmivuo und Plonsey 1995, Abs. 13.4]
[19] siehe [Bayliss 2001, S. 15]
[20] siehe [Nickolls 2002, S. 15]

Abbildung 2.6: Elektroenzephalogramm – gemessen *am* Kopf [Olejniczak 2006, S. 187]

Inversions-problem

Da Neuronenpotentiale im (dreidimensionalen) Gehirn entstehen und EEG diese Realdaten auf das zweidimensionale Elektroenzephalogramm projiziert, ist die genaue Signalquelle im Gehirn nicht direkt aus EEG-Aufzeichnungen abzulesen; die Thematik wird *Inversionsproblem* genannt.[21] In [He et al. 2006][22] werden verschiedene Methoden beschrieben, welche das Inversionsproblem bei EEG lösen.

Frequenzen: α, β, γ, δ, θ

Die in einem EEG-Signal zu beobachtenden Frequenzen sind teilweise benannt und werden nach Tabelle 2.1 in fünf Bereiche eingeordnet:

EEG-Wellenbezeichnung	Frequenzbereich [Hz]	max. Amplitude [μVolt]
Delta (δ)	0.5−4	<100
Theta (θ)	4−8	<100
Alpha (α)	8−13	<10
Beta (β)	13−30 bzw. 14−22	<20
Gamma (γ)	22−30	<2

Tabelle 2.1: EEG: Frequenzen und Amplituden [Malmivuo und Plonsey 1995, Abs. 13.5], [Nickolls 2002, S. 5ff]

Anmerkung β: 14-22 Hz nach [Nickolls 2002, S. 5], jedoch 13-30 Hz nach [Malmivuo und Plonsey 1995, Abs. 13.5]

EEG-Muster je nach Zustand

δ-Wellen werden bei Säuglingen und schlafenden Erwachsenen gemessen oder wenn organische Hirnschäden bestehen; θ-Wellen dagegen bei Kindern und schlafenden

[21] siehe [Olejniczak 2006, S. 186] bzw. [He et al. 2006]
[22] siehe [He et al. 2006, S. 3ff]

Erwachsenen bzw. ebenso bei manchen Erwachsenen während emotionalen Stresssituationen. α-Wellen treten bei *wachen* Personen mit geschlossenen Augen im hinteren Teil des Gehirns (Occipital) auf und β-Wellen herrschen bei normaler Aktivität im Cerebrum (Frontal und Parietal) vor (siehe auch Abbildung 2.2, rechts); γ-Wellen dominieren, wenn Aufmerksamkeit oder eine sonstige sensorische Stimulation besteht. Während intensiver mentaler Anstrengung kann die Frequenz auf bis zu 50 Hz steigen. Beispielhaft können die EEG-Frequenzen in Abbildung 2.7 (links) betrachtet werden.[23]

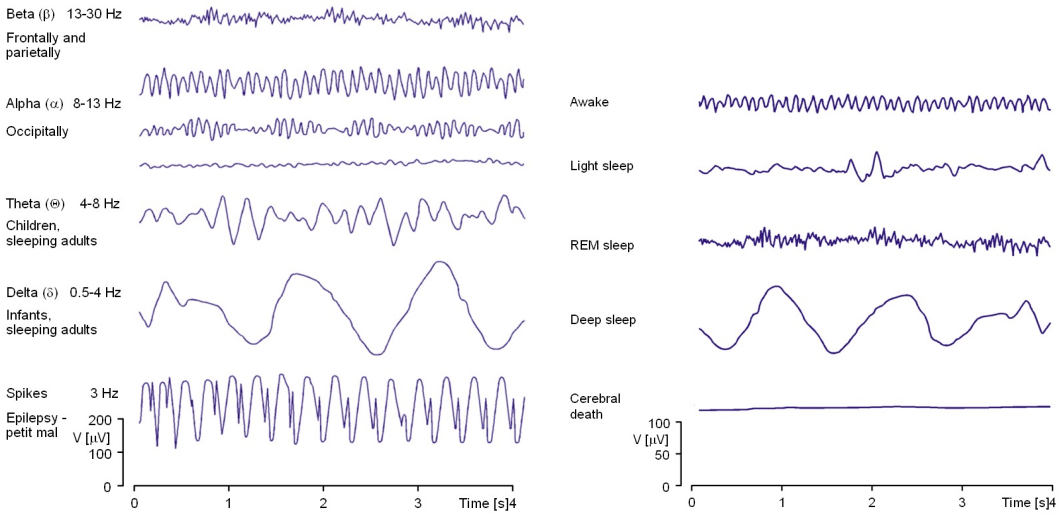

Abbildung 2.7: EEG-Signalmuster [Malmivuo und Plonsey 1995, Abb. 13.5] (links), [Malmivuo und Plonsey 1995, Abb. 13.6] (rechts)

Die Form des EEGs ist also abhängig von den aktuellen Tätigkeiten einer Person (siehe auch Abbildung 2.7, rechts). Mit zunehmender Aktivität verkleinern sich die Amplituden und erhöhen sich die Frequenzen. Wie schon erwähnt, treten bei geschlossenen Augen Alpha-Wellen in den Vordergrund. Während des Schlafes beginnen die Frequenzen zu fallen, bis in einer Tiefschlafphase Delta-Wellen vorherrschen; Amplituden sind hoch.[23]

Eine Zwischenphase des Schlafes bezeichnet man als *Rapid Eye Movement (REM)*-Schlaf. Diese ist, entgegen der Tiefschlafphase, gekennzeichnet durch kleine Amplituden und hohe Frequenzen, ähnlich den Wachphasen. In der REM-Phase gehen intensive Träume vor sich und schnelle Augenbewegungen sind zu beobachten.[24]

REM-Phase

Keine EEG-Signale erscheinen jedoch nach dem *Hirntod*.[25]

Hirntod⇝
Nulllinie

[23] siehe [Malmivuo und Plonsey 1995, Abs. 13.5] bzw. [Nickolls 2002, S. 5ff]
[24] siehe [Nickolls 2002, S. 8]
[25] siehe [Malmivuo und Plonsey 1995, Abs. 13.6]

3 Brain-Computer-Interfaces

Ursprung in Medizintechnik

Traditionell finden *Biosignal*[26]-verarbeitende Systeme im medizinisch-diagnostischen Bereich (z. B. EEG – siehe Abs. 2.2) Verwendung. Dementsprechend sensibel, präzise und damit teuer sind diese Geräte. Die nicht-medizinisch bedingte Anschaffung geschieht daher üblicherweise im Bereich von Forschungseinrichtungen o. ä.

Die Forschungstätigkeiten umfassen zu einem immer größer werdenden Teil die Nutzung der Hirnaktivität zur nicht-manuellen Steuerung von Geräten, insbesondere Computern. In diesem Zusammenhang fallen die Begriffe *Brain-Computer-Interface (BCI)* oder allgemeiner – *Brain-Machine-Interface*. Die involvierten Institutionen können mittlerweile respektable Ergebnisse vorweisen. Die Steuerung von kompletten Computerprogrammen, Maschinen oder generell Eingabegeräten sind mit ausreichendem Training der Probanden ohne weiteres möglich[27].

EEG-Einsatz

Die *Elektroenzephalografie (EEG)* ist, wie schon erwähnt, die kostengünstigste Methode zur Messung der Hirnströme. Die potentielle Mobilität sowie die vielfältig möglichen Versuchsanordnungen sind weitere Gründe, warum vorrangig EEG-Systeme für BCI-Anwendungen zum Einsatz kommen.

rege Forschung

Auf der Webseite der Firma g-tec[28] finden sich eine Vielzahl an Informationen zu Forschungseinrichtungen[29] für BCI, wichtige Veröffentlichungen[30] zum Thema, Internationale Projekte[31] sowie eine Erläuterung zum *Stand der Wissenschaft* allgemein[32].

[26] jegliche Art von Signalen, welche im menschlichen Körper entstehen (z. B. Neuronenpotentiale)
[27] siehe z. B. `http://bci.tugraz.at/downloads.html` (Stand Sept. 2009)
[28] `http://www.gtec.at` (Stand Sept. 2009)
[29] Forschungseinrichtungen: `http://www.gtec.at/research/bci_labs.htm` (Stand Sept. 2009)
[30] Veröffentlichungen: `http://www.gtec.at/research/bci-publications.htm` (Stand Sept. 2009)
[31] Projekte: `http://www.gtec.at/research/Projects.htm` (Stand Sept. 2009)
[32] Stand der Forschung: `http://www.gtec.at/research/bci_research.htm` (Stand Sept. 2009)

3.1 Problemstellung der BCI

Brain-Computer-Interfaces versuchen charakteristische Hirnsignalmuster des Benutzers einer definierten Tätigkeit zuzuordnen (*Klassifikation*), welche daraufhin vom Gerät ausgeführt wird.

Klassifikationsaufgabe

Die Verwendung solcher Systeme kann vorteilhaft sein, wenn die Benutzung der Hände schwierig, eine Interaktion zur Steuerung eines Gerätes aber notwendig ist; sei es eine Verletzung oder Behinderung, welche den Ausschluss der Hände bewirkt oder ein anderweitiger Einsatzbedarf dieser (z. B. im Zuge von manuellen Arbeiten). Nicht verwunderlich ist daher auch das schon lange bestehende Interesse der Militärs an BCI-Systemen.[33]

nicht-manueller Charakter

Militär

Artefakte

Störungen hinsichtlich der reinen EEG-Signale werden *Artefakte* genannt. Sie können entstehen durch physiologische Einflüsse wie *Bewegungen der Augen* (Elektrookulografie Signal [EOG]; siehe Abbildung 3.1), *Herz* (Elektrokardiografie Signal [EKG]) bzw. *Muskeln*[34] (Elektromyografie Signal [EMG]). Weiters bilden sich Artefakte durch andere Einflüsse wie die *Frequenz des Stromnetzes* (Europa: 50 Hz) oder die Veränderung des *Widerstandes der Elektroden* zufolge Schweißbildung unter den Elektroden. EOG-Artefakte zeigen normalerweise hohe Amplituden und werden z. B. vom Blinzeln der Augen ausgelöst. EOG-Aktivitäten treten bei Frequenzen bis zu 4 Hz in Erscheinung.[35]

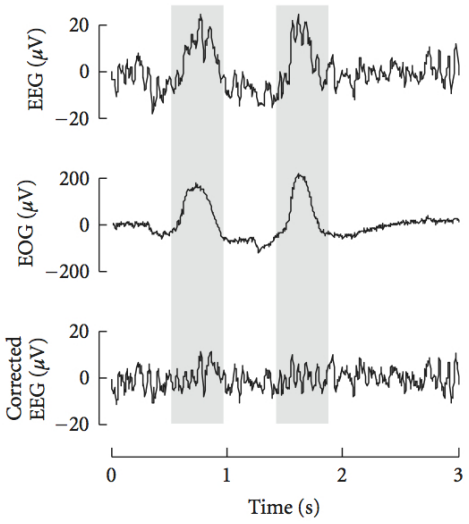

Abbildung 3.1: Artefaktinterferenzen [Scherer et al. 2007, S. 7]

EOG

EKG

EMG

Signalinterferenzen

EMG-Aktivitäten, verursacht durch z. B. Bewegung von Kopf, Kiefer, Zunge oder Körper, liegen im Frequenzbereich über 30 Hz. Anstrengende Tätigkeiten verursachen

[33] siehe [Nelson et al. 1997, S. 30]

[34] Muskeln erhalten ihre „Einsatzbefehle"/*Stellwerte* (wie bereits erklärt) über das Nervensystem, weshalb ebenso elektrische Potentiale auftreten. [Malmivuo und Plonsey 1995, Abs. 2]

[35] siehe [Fatourechi et al. 2007, S. 481ff] bzw. siehe [Nickolls 2002, S. 11]

daher oft EMG-Artefakte, weil dabei Gesichtsmuskelkontraktionen zu erwarten sind.[36]

Artefakte bewirken also eine Interferenz mit den EEG-Daten und können den Klassifikationsvorgang von BCI-Systemen so verfälschen, dass das BCI in Folge die Intention des Benutzers nicht adäquat widerspiegelt. In Hinsicht auf die Nutzbarkeit und Zuverlässigkeit von Brain-Computer-Interfaces müssen somit Vorkehrungen für die Artefakt-Behandlung getroffen werden.

Artefakte⇝ Klassifikation?

Bayliss bezeichnet Systeme, welche EEG und Artefakte zur Klassifizierung heranziehen als *Brain-Body-Actuated-Systems*.[37]

3.2 High Cost BCI-Systeme

Die Firma *g-tech – Guger Technologies OEG*[38] ist ein kommerzieller Anbieter von biomedizinischen Geräten, die nicht nur den medizinischen Bereich abdecken sollen, sondern auch bei vielen Forschungsprojekten im Einsatz sind. g-tech stellt flexible Hard- und Softwarelösungen zur Verfügung, welche eine Basis für alle denkbaren Applikationen und Versuche mit Biosignalen sein können.

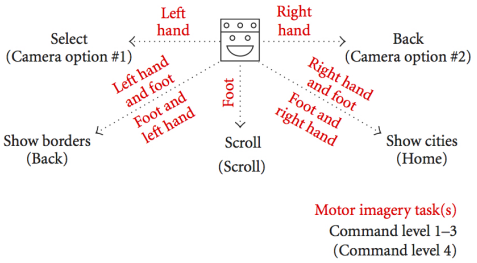

Abbildung 3.2: Graz-BCI: Zuweisung Imagination zu Aktionen [Scherer et al. 2007, S. 6]

Eine Einrichtung, welche u. a. Produkte von g-tech verwendet, ist das *Institut für Semantische Datenanalyse / Knowledge Discovery*[39] an der TU Graz[40]. Im dortigen *Laboratory of Brain-Computer Interfaces* wurde z. B. das *„self-paced 3-class Graz brain-computer interface“* entwickelt. Dieses baut auf ein EEG-System mit drei bipolaren Kanälen (*wenige EEG-Kanäle erhöhen die praktische Verwendbarkeit, wegen der sinkenden Kosten bzw. Vorbereitungs-/Montagezeit*).[41]

Artefakte werden weitestgehend zu reduzieren versucht (z. B. EOG: -80 %) und die Steuerung erfolgt durch *Imagination* von motorischen Handlungen (Bewegungen). Um

Motor-imagination Freeze Mode

[36] siehe [Fatourechi et al. 2007, S. 482]
[37] siehe [Bayliss 2001, S. 8]
[38] `http://www.gtec.at` (Stand Sept. 2009)
[39] `http://bci.tugraz.at` (Stand Sept. 2009)
[40] `http://www.tugraz.at` (Stand Sept. 2009)
[41] siehe [Scherer et al. 2007]

Fehlsteuerung durch Artefakte zu verhindern (*Robustheit*), besteht die Möglichkeit, nach einer Artefakterkennung das BCI zu blockieren („Freeze Mode“).[41]

Sensorischer / Motorischer Cortex

Die am *Motorischen* und *Sensorischen Cortex* (siehe Abbildung 2.2) angebrachten Elektroden können sowohl die Artefakte (EOG, ungewollte Muskelbewegungen – EMG), als auch die gedachten Bewegungen registrieren. Ein Klassifikationsalgorithmus[42] ermöglicht die Erkennung von bestimmen Hirnsignalmustern, woraufhin das BCI zugewiesene Aktionen auslöst (siehe Abbildung 3.2). Nach ausreichendem Training (über *Feedback*-Methoden) waren Versuchspersonen in der Lage, auch komplexe Navigationsaufgaben (siehe z. B. Abbildung 3.3[43]) äußerst befriedigend durchzuführen.[41]

Feedback-Gedanke

Abbildung 3.3: Graz-BCI: Navigation in virtuellen Welten

persönliche Eignung

Es bleibt aber anzumerken, dass gute Ergebnisse nicht von jedem Benutzer (schnell) zu erreichen sind. In [Pregenzer und Pfurtscheller 1999, S. 413] wurden z. B. nach Eingangstests von zehn Probanden nur drei für die tatsächlichen Versuche ausgewählt, da die anderen zu hohe Fehlerraten produzierten (>35%). Als Begründung gaben die Autoren an, dass hohe Fehlerraten zu Demotivation während der Tests führen würden und daher nicht zielführend erschienen. Eine Senkung der Fehler durch weiterführendes Training dieser physiologisch weniger geeigneten Personen wird vermutet, wurde aber im Zuge dieser Studie nicht nachgewiesen.

Es zeigt sich, dass auch hochpreisige BCI-Systeme keineswegs problemlos sind. Als Schwachpunkte können folgende Punkte identifiziert werden:

- Die Abhängigkeit von den physiologischen Anlagen des Benutzers. Viele Personen sind gar nicht fähig, ein EEG-BCI zu bedienen bzw. benötigen vermutlich eine lange Lernzeit.

[42] *Distinction Sensitive Learning Vector Quantization (DSLVQ)* – siehe [Pregenzer und Pfurtscheller 1999]
[43] Bild: `http://bci.tugraz.at/title_img.jpg` (Stand Sept. 2009)

- Die relativ lange Vorbereitungszeit bis das System einsatzbereit ist. Je mehr Elektroden verwendet werden, desto länger dauert die Montage durch Positionsfindung sowie Impedanzkontrolle (Elektroden-Gel).
- Die derzeit noch hohen (Hardware-)Kosten der EEG-BCI-Geräte (je mehr Elektroden, desto teurer).
- Durch die medizinisch übliche stationäre Verwendung ist EEG-Hardware eher nicht für mobile Einsätze ausgelegt (Größe, Empfindlichkeit). Prinzipiell sollte die „Mobilisierung" der Hardware jedoch möglich sein.

3.3 Low Cost BCI-Systeme

„leistbar" ⇝ Massenmarkt

Spielezubehör

Steuerungs-*unterstützung*

Anwendungen im täglichen Leben sind nur durchzusetzen, wenn sich die Kosten für solche Geräte im Rahmen halten. Es sind bereits Produkte erhältlich, welche diverse Möglichkeiten der Biosignal-Verarbeitung auszunutzen versuchen und zugleich preislich attraktiv bleiben. Anfang 2008 wurde z. B. von der Firma *OCZ Technology Inc.*[44] der *Neural Impulse Actuator (NIA)* (siehe Abbildung 3.4[45]) vorgestellt. Ein Gerät welches verspricht, die Benutzung der Maus/Tastatur in Computerspielen durch *Biosignale* zwar nicht gänzlich ersetzen zu können, sehr wohl aber als Ergänzungshardware das Spieleerlebnis entscheidend zu verbessern. Die Biosignale sollen im Vergleich zu konventionellen Eingabemethoden eine um 50% verringerte Reaktionszeit aufweisen.[46]

Abbildung 3.4: Der NIA von OCZ Technology

Mit ca. 100 €[47] befindet sich dieses BCI, bezüglich der Anschaffungskosten, etwa um den Faktor 20 unter den medizinisch verwendeten Einstiegsgeräten und würde damit prinzipiell vielfältige Chancen im Massenmarkt bieten. Der NIA wird von OCZ jedoch als Spielehardware verkauft, weshalb die inkludierte Software für Nicht-Spiele-Anwendungen wenig geeignet ist. Im Wesentlichen können damit für verschiedenste Spiele eigene Profile definiert werden, welche Biosignal-Schwankungen auf Tastatur- oder Mausein-

[44] `http://www.ocztechnology.com` (Stand Sept. 2009)
[45] Bild: `http://www.ocztechnology.com/images/products/accessories/b/NIA_headband_1_big.jpg` (Stand Sept. 2009)
[46] `http://www.ocztechnology.com/products/ocz_peripherals/nia-neural_impulse_actuator` (Stand Sept. 2009)
[47] siehe `http://geizhals.at/a328571.html` (Stand Sept. 2009)

gaben zuweisen. OCZ stellt (derzeit)[48] auch keine Programmierschnittstelle (SDK) zur Verfügung, die natürlich eine Erweiterung der Möglichkeiten böte.

keine offizielle SDK

Ähnliche – und in Konkurrenz zum NIA stehende – Produkte sind der *EPOC* (siehe Abbildung 3.5, links[49]) von *Emotiv Systems*[50]. Für dieses BCI ist mittlerweile auch eine kostenpflichtige SDK[51] erhältlich.

Konkurrenz-systeme

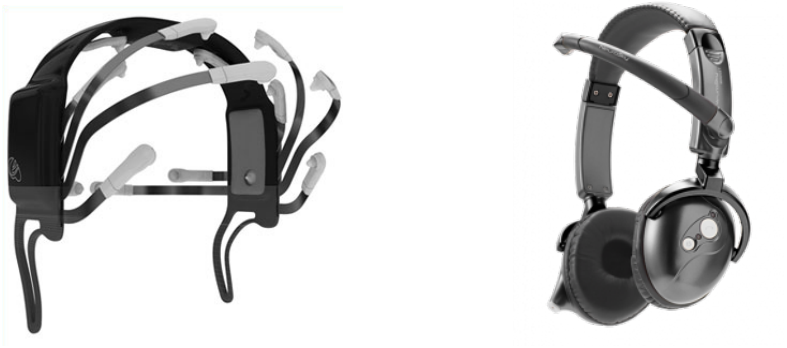

Abbildung 3.5: Der EPOC von Emotiv Systems (links); der MINDSET von NeuroSky (rechts)

Außerdem existiert der *MindSet* (siehe Abbildung 3.5, rechts[52]) von *NeuroSky*[53] inkl. kostenpflichtiger SDK[54]. Beiden Geräten ist gemeinsam, dass sie noch etwa 1.5 bis 2 mal teurer als der NIA sind und die SDKs – je nach Einschränkung des Zugriffes auf die Messdaten – bis zu 10.000 $[55] kosten.

Der NIA steht dem *Institut für Bauinformatik*[56] der TU Graz zur Verfügung und wird derzeit im Zuge anderer Forschungsarbeiten evaluiert. Das Interessante am NIA stellt die technologische Nähe zu einem im medizinisch-therapeutischen Bereich jahrelang erprobten Gerät dar – dem *Cyberlink Brainfingers*-System.

Versuche mit NIA

Basissystem *BAT Cyberlink*

[48] Lizenzschwierigkeiten, jedoch in Entwicklung lt. `http://cerebralhack.com/2009/07/cerebralhack-at-ocz-hq-for-an-exclusive-interview-with-dr-michael-schuette` (Stand Sept. 2009)
[49] Bild: `http://digitalventurepartner.com/blog/wp-content/uploads/2009/03/emotiv-headset.png` (Stand Sept. 2009)
[50] `http://www.emotiv.com` (Stand Sept. 2009)
[51] siehe `http://www.emotiv.com/developers.html` (Stand Sept. 2009)
[52] Bild: `http://cdn.shopify.com/s/files/1/0031/6882/products/1415878_1806_small.png?1249502826` (Stand Sept. 2009)
[53] `http://www.neurosky.com` (Stand Sept. 2009)
[54] siehe `http://company.neurosky.com/developers` (Stand Sept. 2009)
[55] Stand Sept. 2009
[56] `http://bauinformatik.tugraz.at` (Stand Sept. 2009)

Die Technologie des NIAs lizensiert OCZ nämlich von der Firma *Brain Actuated Technologies Inc. (BAT)*[57], welche den Cyberlink herstellt. Sowohl die Hardware, als auch die Software des NIAs sind Abwandlungen dieses Systems. Der Cyberlink kostet über 2.100 $[58], also mehr als das 14-fache[59] gegenüber dem NIA. Eine Programmierschnittstelle ist ebenso nicht erhältlich, andere Softwareanwendungen können mit dem System jedoch angesprochen werden („Launch wizard“).

militärische Basis

Die Grundlagen zum *Cyberlink Brainfingers* entstanden im militärischen Bereich[60]. Maßgeblich beteiligt an der Entwicklung war Andrew Junker, welcher auch Gründer von BAT ist.

Patente

Idee: eigene Software

Aus Junkers Patenten[61] [62] [63] mit zahlreichen technischen Details zum Cyberlink-System wird deutlich, dass der NIA überaus ähnlich aufgebaut ist. Dies und weitergehenden Arbeiten am *Institut für Bauinformatik* lassen vermuten, dass wahrscheinlich nur die Software den entscheidenden Unterschied zwischen NIA und Cyberlink ausmacht. Eine eigene Softwareentwicklung für den NIA könnte das, in einigen Studien[64] zum Cyberlink bestätigte, Potential also freilegen.

[57] `http://www.brainfingers.com` (Stand Sept. 2009)
[58] siehe `http://www.brainfingers.com/price.htm` (Stand Sept. 2009)
[59] Wechselkurs: 1 $ = 0,68 € (Stand Sept. 2009)
[60] siehe [Junker et al. 1989] bzw. siehe [Nelson et al. 1997]
[61] siehe [Junker 1993]
[62] siehe [Junker 1995]
[63] siehe [Junker und Berg 2001]
[64] siehe [Nelson et al. 1997]; [Marler 2004] und [Junker et al. 2008a] bzw. [Junker et al. 2008b]

4 OCZ NIA – Die Hardware

Der *Neural Impulse Actuator (NIA)* von OCZ Technologies Inc. ist lt. Handbuch

> *„ein neuartiger Ansatz der Computerinteraktion über Biosignale, genau genommen elektrische Potenziale, die über die Stirn des Benutzers aufgenommen werden. Zu den Biopotenzialen gehören Elektromyogramm, Elektroenzephalogramm und Elektrookulogramm. All dies sind elektrische Signale, die jeweils durch Aktivitätsmuster in Muskeln, Gehirn und Augen erzeugt werden. Diese Potenziale werden verstärkt und in unterschiedliche Frequenzkomponenten aufgespaltet, um einzelnen Kanälen unterschiedliche Befehle zuzuteilen.“*[65]

4.1 Überblicksmäßige Beschreibung

Biopotentiale

reaktions-schnell

Der NIA ist ein Computer-Eingabegerät aus der Spielebranche, welches verspricht, durch Erkennung von elektrischen Potentialänderungen am Kopf des Benutzers, die Steuerung von Computerspielen zumindest teilweise übernehmen zu vermögen. Da der Spielerfolg während der Anwendung oft sehr abhängig von schnellen Reaktionen des Spielers ist, erscheint die direkte Übertragung von Befehlen an den Computer vorteilhaft zu sein; die natürliche Trägheit der Muskelaktivierung entfällt nämlich.

Abbildung 4.1: Der NIA in Verwendung

Der NIA besteht aus einer Interface-Box mit Aluminiumgehäuse, welche drei – an einem Stirnband befestigte – Elektroden über Kabel mit dem Computer (USB[66]-Schnittstelle) verbindet (siehe Abbildung 4.1). Eine eigene wird nicht benötigt. (siehe Abbildung 3.4)

Basissystem *BAT Cyberlink*

Die Technologie des NIAs wird OCZ von der Firma *Brain Actuated Technologies Inc. (BAT)*[67] zur Verfügung gestellt. Sowohl die Hardware, als auch die Software sind Abwandlungen des *Cyberlink Brainfingers* Systems von BAT, welches vorrangig im medizinisch/therapeutischen Bereich bei Personen mit Behinderungen Verwendung findet. Das Cyberlink-System kostet über 2.100 $[68]. Eine Programmierschnittstelle ist nicht erhältlich, andere Softwareanwendungen können mit dem System jedoch ebenso angesprochen werden („Launch Wizard“).

4.1.1 Die Basis: Cyberlink Brainfingers™

Abbildung 4.2: Das Cyberlink-Schema

nicht-manuelle Interaktion

Der Cyberlink erlaubt die Benutzung und die Kommunikation mit Computern ohne dem Erfordernis manueller Tätigkeit. Das System besteht aus einer externen, batteriebetriebenen Interface-Box mit Kabelverbindung zu einem Kopfband (siehe Abbildung 4.2[69]). Das Interface wird über die *USB*- oder eine *serielle*[70] Schnittstelle mit dem Computer verbunden.[71]

Interface-Box

Kopfband
3 Elektroden

Das Kopfband hält im Bereich der Stirn drei austauschbare Elektroden, welche elektrische Potentiale über den Hautkontakt messen. Als Elektroden sind von BAT *Trockensensoren* (langlebig; Verwendung mit separat erhältlichem Feuchtigkeitsgel) oder *Gel-Sensoren* (keine extra Hautbehandlung mit Gel erforderlich; für Situationen, wo starke Kopfbewegungen zu erwarten sind – „Verrutschgefahr“) zu beziehen.[71]

Artefaktnutzung

Brainfingers
Digitalschalter

Die aufgenommenen Signale sind im Wesentlichen auf Augenbewegungen (EOG), Gesichtsmuskeln (EMG) und Hirnströme (EEG) zurückzuführen. Die Hard- und Softwarekombination leitet aus dieser Summe an elektrischen Potentialdaten elf getrennte Kanäle ab, welche BAT *„Brainfingers“* nennt. Sie können als simple Ein/Aus-Schalter mit Schwellwert fungieren bzw. können kontinuierlich „verwertbare“ Biosignale für „Joysticks“ verwendet werden. Über Joysticks definierte Aktionen werden im Computer ausgelöst, wenn sich die betreffenden Biosignale in gewissen Bereichen befinden.[72]

analoge Joysticks

[65] siehe [OCZ 2008, S. 1]
[66] `http://www.USB.org` (Stand Sept. 2009)
[67] `http://www.brainfingers.com` (Stand Sept. 2009)
[68] siehe `http://www.brainfingers.com/price.htm` (Stand Sept. 2009)
[69] `http://www.brainfingers.com/images/cyberl3.gif` (Stand Sept. 2009)
[70] siehe [Bra 2004, S. 7]; Seriell: RS232 [eia 1969]; USB: über Adapter
[71] siehe [Bra 2004, S. 8]
[72] siehe [Bra 2004, S. 7ff] bzw. `http://www.brainfingers.com/technical.htm` (Stand Sept. 2009)

Wie die Kanäle mit dem Computer benutzt werden, bleibt für den Benutzer frei konfigurierbar; wie später noch gezeigt wird, müssen diese Einstellungen oftmals auf die Person angepasst werden, um eine befriedigende Benutzerinteraktion mit dem System zu erlangen.[72]

Flexibilität

Dem wörtlichen Sinne eines Brain-Computer-Interfaces entspricht also weder der Cyberlink Brainfingers noch der OCZ NIA. Es werden nämlich nicht reine Gehirnsignale (EEG) ausgewertet werden, sondern auch – und sogar zu einem größeren Teil – Muskel (EMG)- und Augenbewegungspotentiale (EOG). Obwohl an der Stirn durchaus EEG-Daten (μVolt-Bereich) aufgenommen werden können, treten dort Artefakte[73] sind die Biopotentiale um ein Vielfaches stärker. Passender wäre also eine Beschreibung als *Brain-Body-Actuated*-System, welches hauptsächlich Muskelsignale verwertet.[74]

Biopotentiale

Im Folgenden werden zur Vereinfachung der Terminologie sowohl Brain-Computer-Interfaces, wie auch Brain-Body-Actuated-Systems mit dem Begriff *BCI* abgekürzt.

4.1.1.1 Technologie des Cyberlink Systems

Die an der Stirn gemessenen Potentialdaten werden von BAT in drei Frequenzbänder eingeteilt:[75]

Frequenzanalyse: 11 Brainfingers

- Die **EOG**-Signale treten im Bereich von 0.2 bis 3.0 Hz auf und sind z. B. für links/rechts-Bewegungen des Mauszeigers zu gebrauchen.

  EOG 0.2−3 Hz

- Der zweite Frequenzbereich (**EEG**) liegt zwischen 0.5 und 45 Hz. Dieser wird mit einem patentierten Verfahren[76] in weitere zehn Frequenzbänder untergliedert. Diese sind um die Frequenzen 0.95, 2.75, 4.40 – 7.75, 9.50, 11.45 –13.25, 16.50, 21.20 und 25.00 Hz zentriert. Bei BAT heißen diese Frequenzbänder F1–F10, wovon F1–F3 Augenbewegungen widerspiegeln und F7–F10 Gesichtsmuskelkontraktionen anzeigen. Wie breit die Frequenzbänder sind, ist vom Benutzer festzulegen und regelt die Reaktionsgeschwindigkeit der jeweiligen Brainfingers[77].

  EEG 0.5−45 Hz

- Zwischen 70 und 1000 Hz befindet sich der dritte Kanal des Cyberlinks und dieser entspricht einem **EMG**-Signal. Der Bereich lässt sich sehr gut für analoge Steuerungen einsetzen, bietet hohe Genauigkeit (>90 %) und schnelle Reaktionszeiten (ca. 0.2 Sek.).

  EMG 70−1000 Hz

[73] siehe Abs. 3.1, S. 13
[74] siehe [Bayliss 2001, S. 3 und 9]
[75] siehe [Junker et al. 2008a, S. 2]
[76] siehe [Junker 1995]
[77] siehe [Junker 1995, 2]

Militär

Die Grundlagen zum *Cyberlink Brainfingers* System entstanden im militärischen Bereich[78]. Maßgeblich an der Entwicklung beteiligt war Andrew Junker, welcher auch Gründer des Herstellerunternehmens *Brain Actuated Technologies Inc. (BAT)* ist.

Patente

In seinen Patenten[79] [80] [81] mit zahlreichen technischen Details zum Cyberlink-System arbeitet Junker heraus, dass eine der wichtigsten Eigenschaften des Cyberlink die Verwendung einer Kombination von EEG und EMG ist. EMG ist bei der Messung an der Stirn eine Summe der Muskelgruppen von *Augen, Nacken, Kiefer, Stirn („Frontalis"-Muskel)* etc. Darüber hinaus unterscheiden sich die Signal-Frequenzen von 0.5-45 Hz, je nach aktivierter Muskelgruppe, und können so bewusst gesteuert werden. Zudem unterstützt die Software den Benutzer über Feedback-Methoden, welche das Erlangen der Kontrolle erleichtern sollen.

Feedback

Lock-In Verstärker

Junker beschreibt, dass über einen eingebauten *„digitalen Lock-In Verstärker"* schnell saubere (hohes *Signal/Noise-Verhältnis*) Amplituden der gewählten Brainfingers-Frequenzen gewonnen werden können. Alternativ dazu wäre für diese Frequenzanalyse der Potentialdaten der Einsatz einer *Fourier Transformation (FT)* möglich. Die im Cyberlink eingebaute *Lock-In*-Vorgangsweise ist nach Junker leicht auf andere Anwendungsgebiete übertragbar, wenn – wie im Falle von Biosignalen – Eingabedaten mit kleinen Signal/Noise-Verhältnissen vorherrschen.[79]

bei kleinen Signal/Rausch-Verhältnissen

große Elektroden

Weiters wird in [Junker 1995] erklärt, dass die verwendeten Elektroden größer als EEG-Elektroden sein sollen, um genügend Signale aufnehmen zu können (etwa ähnlich den EKG-Elektrodenabmessungen; trotz des Verlustes von spatialer Auflösung). Zum Betrieb des Cyberlink-Systems ist die Anbringung von Elektroden prinzipiell an allen Muskeln des Körpers möglich, da Muskelsignale die dominierende Quelle der Steuerung sind. EEG-Signale können jedoch nur am Kopf aufgenommen werden und wegen der praktisch einfachen Handhabung wird beim Cyberlink das Kopfband verwendet.

4.1.1.2 Anwendungsergebnisse

Erfolg durch Software

Das Cyberlink-System bietet eine umfangreichere Software als der NIA. Vor allem fehlen dem NIA viele Lernprogramme, welche es dem Benutzer erleichtern, das Gerät

[78] siehe [Junker et al. 1989] bzw. siehe [Nelson et al. 1997]
[79] siehe [Junker 1993]
[80] siehe [Junker 1995]
[81] siehe [Junker und Berg 2001]

besser zu kontrollieren[82]. BAT richtet den Fokus klar in den Bereich der allgemeinen Computersteuerung und kann dabei auch beachtliche Erfolge vorweisen.

Studien

Neben den in [Nelson et al. 1997] gezeigten Vorteilen bei der unterstützenden Steuerung von Kampfflugzeugen, wurde der Cyberlink vor allem in therapeutischen Situationen getestet.

erhöhte Lebensqualität

So konnten z. B. in einer Brainfingers-Studie über einen Zeitraum von acht Wochen (an 16 verschiedenen Tagen) mit vier schwerst behinderten[83] Schülern ausgezeichnete Ergebnisse herausgearbeitet werden. Die Studienteilnehmer waren innerhalb kurzer Zeit in der Lage, einfache Lernprogramme zu bedienen bzw. erweitert mit ihrer Umwelt zu kommunizieren, was ohne die BCI-Technologie zuvor nicht möglich war.[84]

gleichmäßige Fortschritte

Adaptionen

Die zeitliche Entwicklung der Fähigkeiten im Umgang mit dem Cyberlink ist in Abbildung 4.3 für jeden Schüler (A-D) aufgetragen. Bemerkenswert ist wie gleichmäßig der Lernerfolg dabei voranschritt. Anzumerken ist jedoch, dass, ausgehend von den Standardwerten, für diese guten Ergebnisse individuell verschiedene Kanaleinstellungen/-wechsel vorgenommen werden mussten.[85]

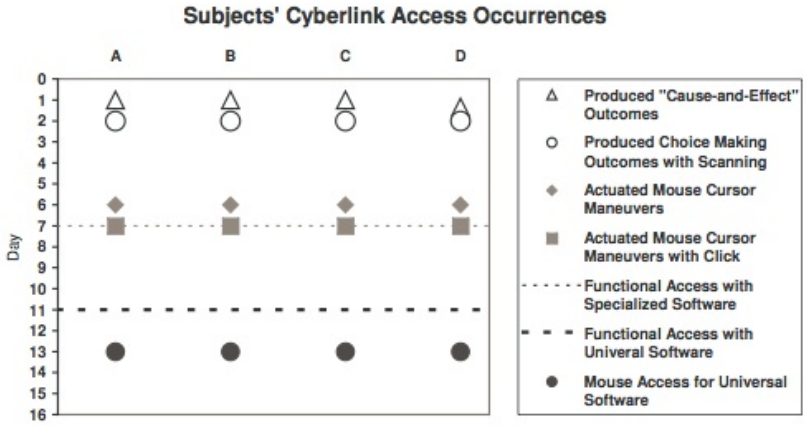

Abbildung 4.3: Entwicklung der Fähigkeiten am Cyberlink Brainfingers System [Junker et al. 2008a, S. 25]

zahlreiche Versuche

Die Resultate einer weiteren Brainfingers-Studie mit 32 Teilnehmern sowie verschiedenen Behinderungen können in [Junker et al. 2008b] nachgelesen werden. Andere Beispiele sind auf der Webseite von BAT unter `http://www.brainfingers.com/success.htm`[86] aufgelistet.

[82] siehe [Bra 2004, S. 15ff] bzw. `http://www.brainfingers.com/cyberlink.htm#F4F6` (Stand Sept. 2009)
[83] siehe [Marler 2004, S. 14]
[84] siehe [Junker et al. 2008a, S. 25]
[85] siehe [Junker et al. 2008a, S. 17]
[86] Stand Sept. 2009

4.2 Lieferumfang

Abbildung 4.4: Die Hardwarekomponenten des NIAs: Interfacebox (links); Kopfband mit drei Elektroden (rechts)

Zum Lieferumfang des NIAs gehören[87]

- das Handbuch,
- die Installations-CD,
- der NIA (siehe Abbildung 4.4, links[88]),
- das Kopfband (siehe Abbildung 4.4, rechts[89]) und
- ein USB-Kabel A-to-B.

Abbildung 4.5: Kopfband-positionierung

flexibles Kopfband

Lt. Handbuch[90] besteht das Kopfband aus Kunststoff, wobei ein verstellbares Gummiband die Anpassung und Fixierung an unterschiedliche Kopfformen und -größen ermöglicht. Im Bereich der Stirn befinden sich drei rautenförmige Sensoren (Elektroden) aus Kunststoff (*„Kohlenstoff-Nanofaser Technologie“*). Zur Verwendung des NIAs soll sich die mittlere Elektrode mittig auf der Stirn und nahe über den Augenbrauen befinden (siehe Abbildung 4.5). Der Elektrodenkontakt zur Haut ist für die Signalweiterleitung essentiell und trockene Haut könnte mit einer Feuchtigkeitslotion vorbehandelt werden.

Elektrodenposition

Hautmilieu

Weitere Details zur Hardware sind vom Hersteller selbst nicht zu erfahren.

[87] siehe [OCZ 2008, S. 1]
[88] Bild: `http://www.ocztechnology.com/images/products/accessories/b/OCZ_NIA_big.jpg` (Stand Sept. 2009)
[89] Bild: `http://www.ocztechnology.com/images/products/accessories/b/NIA_headband_big.jpg` (Stand Sept. 2009)
[90] siehe [OCZ 2008, S. 3]

4.3 Elektronische Bauteile

Abbildung 4.6: Die Bestandteile der NIA-Interfacebox

technische NIA-Details

Aufgrund der spärlichen Aussagen von OCZ zum technischen Aufbau des NIAs haben interessierte Benutzer die Hardware genauer analysiert.

Internetquellen

In einem Internetforum unter `http://www.genmay.com/showthread.php?t=798717`[91] werden Bilder eines zerlegten NIAs gezeigt (siehe Abbildung 4.6) und beschrieben. Ebendort wurde auch der Versuch unternommen, einen Schaltplan der NIA-Elektronik anzufertigen (siehe Abbildung A.1 im Anhang, S. ii).

Interview

In einem Interview[92] beschreibt, der an der Entwicklung des NIAs maßgeblich beteiligte, Dr. Michael Schuette, OCZ Vice President Technology Development[93], ausführlich die Geräteeigenschaften und -hintergründe. Außerdem finden sich auf der Webseite `http://m8ta.com/index.pl?ptags=OCZ`[91] Angaben zur Elektronik des NIAs. Die folgenden Bilder und Beschreibungen der Bauteile stammen unverändert

[91] Stand Sept. 2009
[92] siehe `http://www.lostcircuits.com/mambo//index.php?option=com_content&task=view&id=32&Itemid=47` (Stand Sept. 2009)
[93] siehe `http://www.ocztechnology.com/aboutocz/executive_profiles.html` (Stand Sept. 2009)

von dieser Quelle und konnten im Zuge dieser Arbeit nicht weitergehend überprüft werden. Schlüsse aus der Hardwareanalyse werden für spätere Ausführungen hilfreich sein.

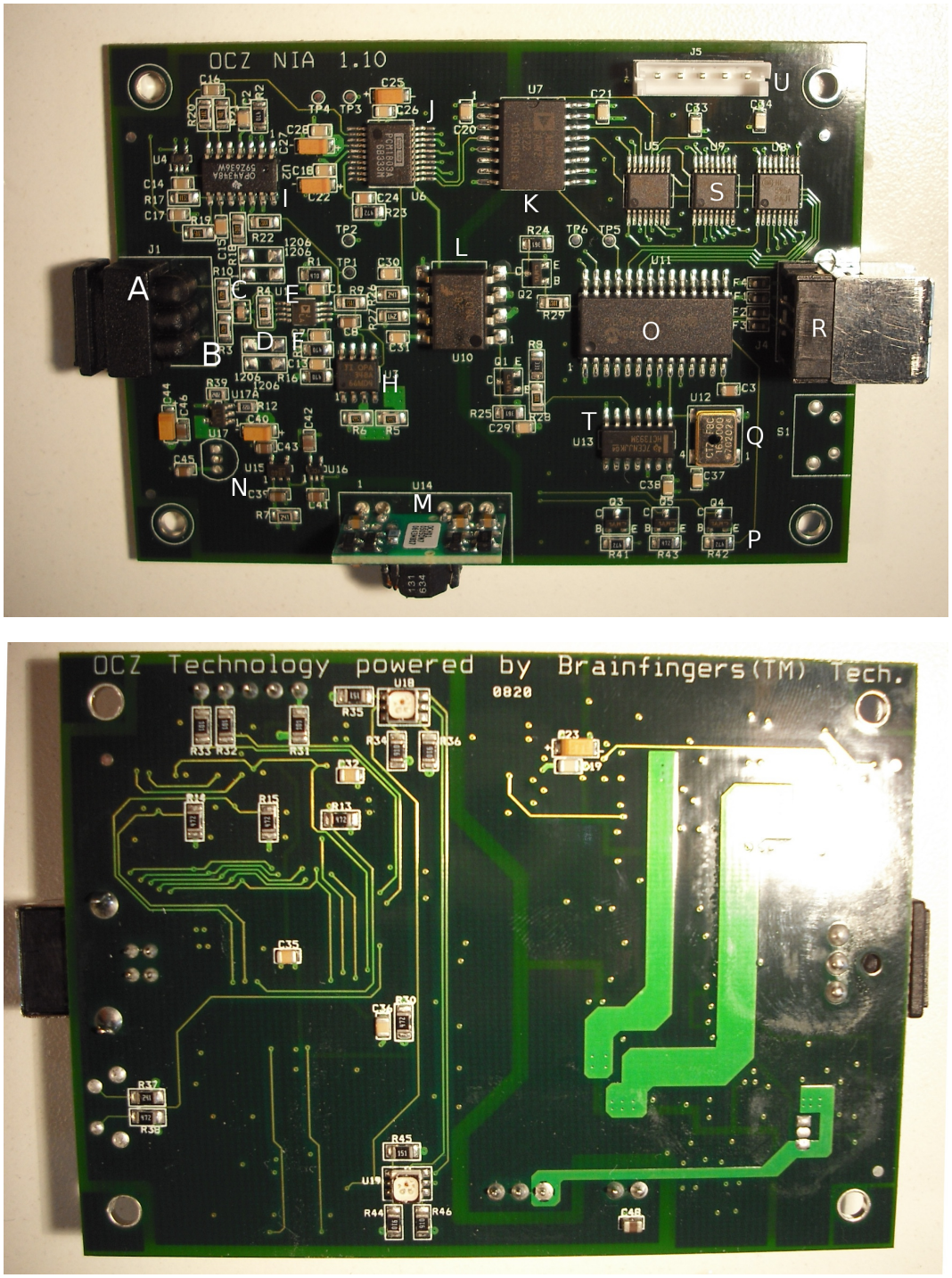

Abbildung 4.7: Die Platine des NIAs

Es folgt eine Beschreibung der in Abbildung 4.7 ersichtlichen Komponenten lt. `http://m8ta.com/index.pl?ptags=OCZ`[91]:

Elektronik-analyse

A *– Input connector. Center channel is isolated ground; outside two channels are the signal. They had to make this custom so people couldn't plug it into other (possibly dangerous) stuff.*

B *– Input current limiting resistors, in series with signal, 4.02K*

C *– Dual capacitor from input channels to shared ground (I think; the cap has 4 contacts, 2 at the end, 2 in the middle; I assume they use this package to get very accurately matched capacitance so as not to hurt the CMRR of the instrumentation amplifier).*

D *– Gain-setting resistor, 1.00K. Sets the instrumentation amplifier gain to 50 (I think).*

I *– do not know what devices were intended for the 1206 footprints above and below this resistor...*

E *– Instrumentation amplifier, Analog Devices logo, AD8220 by my guess, A-grade. Measures the difference in voltage between the two input channels (left and right electrodes on the headband).*

F *– 47 ohm resistors & capacitors to filter the power supply to the instrumentation amplifier.*

H *– Opamp, Texas Instruments OPA348A. Looks like it is used as feedback to the instrumentation amplifier reference pin to effect highpass operation (?).*

I *– Quad opamp, TI OPA4348A. Used to filter the signal; I did not go through the filter topology, but they might have copied it off the AD8220 datasheet ;)*

J *– Stereo ADC, Texas Instruments (Burr-Brown logo, TI bought BB) PCM1803A. Only one channel is used. 24 bits, 96khz max sampling rate; device in master mode (Mode1 = 0V, Mode0 = 3.3v); Fs = SCLK/512 -> sampling rate = 3.90625 KHz.*

K *– Three channel digital isolator, Analog Devices ADUM1300. Transmits the ADC's DOUT, BCK, and LRCLK signals to the USB (non-isolated) side.*

L *– Two-channel optical (?) isolator; unknown type; used to drive the ADC's SCLK and some other signal ? from Joe Pits: „Yeah, optical*

isolator with logic gates for high speed I guess (HCPL2631S). I'm also not sure what the second signal does, it goes to U4 (JSR marking). I suspect it could be a switch which adds C14 + R17 in the feedback loop of U2C (see the schematic). But I don't know what the reason for this is."

M – *Isolated supply daughterboard, Texas Instruments logo, very simple design: driver is 2 BJTs (which get hot!) in push-pull topology; bases are driven by windings on the toroidal transformer; transformer center tap seems to go to USB VCC. Output is +-5V.*

N – *+3.9V, +3.3, and -3.9V power supply circuitry. I cannot identify the SOT-23-5's and SC-70's here.*

O – *PIC18F2455*[94]*, with USB 2.0 (obviously!) SOIC-28 package. device comes up as (on my Linux box, Debian Lenny, kernel 2.6.24):*

```
usb 4-1: new full speed USB device using uhci_hcd and address
8 hiddev96hidraw1: USB HID v1.10 Device
[Brain Actuated Technologies Neural Impulse Actuator Prototype
1.0] on usb-0000:00:1a.1-1
```

I'll put up a usbmon trace later, maybe.

P – *Transistors for driving the tricolor LEDS on the bottom of the board.*

Q – *16.0000 MHz crystal. Needed for correct USB timing; clocks the PIC at 48Mhz.*

R - *USB type B connector. Note the ferrites to the left. (I though they were fuses, but I accidentally shorted Vdd to ground while probing the programming connector, and these let out a little smoke rather than blowing completely. Had they been fuses, they would be open circuit now. This is consistent with Joe Pit's analysis.)*

S – *74HCT595A*[95] *8-bit shift registers, to convert the serial data into parallel data for the PIC to read in. 3 devices = 24 bits in total. Note that the 74HCT595A has a output enable, which permits the PIC to read the 3 bytes of the sample sequentially. Otherwise, as Stefan Jung (via the openeeg-list) points out, the PIC would not have enough data pins (28 pins vs. 24 bits)!*

[94] siehe `http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010273` (Stand Sept. 2009)

[95] siehe `http://www.onsemi.com/pub_link/Collateral/MC74HC595A-D.PDF` (Stand Sept. 2009)

T – *74HCT393, Texas Instruments logo, Dual 4-bit binary ripple counter. Used to drive the ADC with a 2Mhz clock, which puts the sampling rate at (as before) 3.90625 KHz.*

U – *Programming connector. That's right, a programming connector! Looks to be the same as a PIC ICSP*[96] *connector (pointed out on „hack a day"*[97]*) So far as I can tell:*

- *Pin 1 = +5V, PIC pin 1, (through 100 ohm resistor), Vpp (?)*
- *Pin 2 = PIC pin 20 , Vdd*
- *Pin 3 = PIC pin 19 , Vss*
- *Pin 4 = PIC pin 28 (through 100 ohm resistor), PGD*
- *Pin 5 = PIC pin 27 (through 100 ohm resistor), PGC*

I do not know if the device can be reprogrammed, though it looks that way. (from `http://genmay.com/showthread.php?t=798717` *– bootloader (to address 0x07ff) can be read, but everything above that is read-protected)*

Für das Verständnis der vom NIA gelieferten Daten sind folgende Erkenntnisse der vorangegangenen Beschreibung herauszustreichen:

Kernpunkte

- Der Schriftzug auf der Platine des NIAs (siehe Abbildung 4.7) zeigt, wie auch bereits in Abs. 4.1 erklärt, dass die Hardware auf der Brainfingers-Technologie von BAT basiert.

Cyberlink-Technologie

- Die gemessenen Potentialdaten der mittleren Kopfbandelektrode dienen als Bezugswert (*Referenzelektrode*) für die Messungen der linken bzw. rechten Elektrode. (siehe die Beschreibung von *Bauteil A*)

referenzielle Messung

- Die vom NIA gelieferten Daten entsprechen der Potentialdifferenz zwischen den äußeren Kopfbandelektroden. (siehe die Beschreibung von *Bauteil E*)
  Lt. dem offiziellen Forum zum NIA[98] wird eine *Common Mode Rejection* durchgeführt, welche idente Signalkomponenten an der linken und rechten Elektrode beseitigen soll; es bleibt nur ein Unterschiedssignal und dieses wird digitalisiert.[99]

1 Ausgabesignal

- Im NIA werden die gemessenen Daten bereits durch elektronische Filter und

rudimentäre Hardwaresignal-verarbeitung

[96] siehe `http://www.instructables.com/id/Understanding-ICSP-for-PIC-Microcontrollers` (Stand Sept. 2009)

[97] siehe `http://hackaday.com/2008/09/18/ocz-neural-impulse-actuator-teardown` (Stand Sept. 2009)

[98] siehe `http://www.ocztechnologyforum.com/forum/showthread.php?t=60008` (Stand Sept. 2009)

[99] nach `http://www.ocztechnologyforum.com/forum/showthread.php?t=59492` (Stand Sept. 2009) erscheint es wahrscheinlich, dass der Forumsteilnehmer *„Unregistered"* Dr. Michael Schuette, OCZ Vice President Technology Development, ist

Verstärker aufbereitet sowie digitalisiert.

- Der analoge Teil der NIA-Platine, welcher mit dem Kopfband verbunden ist, wird durch einen *Optokoppler* (siehe die Beschreibung von *Bauteil L*) vom digitalen Teil („Computer-Seite“) getrennt. Dies sind rechtlich bedingte Sicherheitsvorkehrungen zur galvanischen Trennung des Menschen vom Stromnetz. Die Biopotentialdaten an den Elektroden werden *passiv gemessen*.[100]

Trennung Mensch/Stromnetz

- Der NIA liest die analogen Potentialdaten an der Stirn kontinuierlich ein und digitalisiert sie ca. 3 906 mal pro Sekunde („Sampling-Rate = 3.906 kHz“). (siehe die Beschreibung von *Bauteil J* und *T*)

Sampling-Rate: 3.906 kHz

- Der NIA produziert jeweils Daten mit einer Größe von 24 Bit, welche jedoch in drei 8 Bit-Werte zerlegt und seriell an die USB-Schnittstelle gesendet werden. (siehe die Beschreibung von *Bauteil J* und *S*)

Im Forum wird bestätigt, dass OCZ bereits an einem Nachfolgemodell arbeitet. Die herausstechenden Merkmale des neuen Gerätes sind seine *Kabellosigkeit* und eingebaute Beschleunigungssensoren.[101]

Nachfolger: kabellos sowie Inertialsensoren

[100] siehe `http://www.ocztechnologyforum.com/forum/showthread.php?t=60008` (Stand Sept. 2009)
[101] siehe `http://www.ocztechnologyforum.com/forum/showthread.php?t=61097` (Stand Sept. 2009)

5 OCZ NIA – Die Software

Spielefokus

Die Philosophie von OCZ ist, dass der Benutzer die Kontrolle des NIAs erlernt, indem er (seine bevorzugten) Computerspiele spielt. OCZ erklärt in [OCZ 2008, S. 11] die angestrebte Vorgangsweise:

> *„[...] muss der Benutzer einige der erlernten Gewohnheiten vergessen und sich einen neuen Satz von Fähigkeiten angewöhnen, um mit dem Computer interagieren zu können.“*

Anpassung an den NIA

Ein Anwender des NIAs muss daher wahrscheinlich einige seiner vormaligen Spielweisen ablegen, kann dafür aber eine Vielzahl neuer Fertigkeiten erlernen. Der Benutzer hat sich also der Funktionsweise des Gerätes anzupassen.

implizites Lernen

Während der ersten Minuten im Spiel mit dem NIA soll der Anwender einfach beobachten; zusehen wie sich das Spiel, gesteuert über den NIA (wenn auch noch unbewusst), verhält. OCZ geht davon aus, dass das Gehirn aus den Beobachtungen Rückschlüsse zieht und mit der Zeit eine bewusste Steuerung möglich wird. Außerdem behauptet OCZ, dass Anfänger üblicherweise die Muskelsignale einsetzen, sie später aber auf feinere, schnellere EEG-Signale umzusteigen in der Lage sind.[102]

Feedback?

Der Ansatz von OCZ mag für Spieler „mühelos“ erscheinen, da sie während der Lernphase ihrem Hobby nachgehen. Spielerisch nicht interessierte NIA-Benutzer werden die Vorgangsweise allerdings als nicht befriedigend ansehen; sie wollen nämlich nur den NIA erlernen, nicht aber stundenlang ohne Feeback/„Fortschrittsanzeige“ an Computerspielen „probieren“.

5.1 Die mitgelieferte NIA-Original-Software

Windows-Software

Die auf CD mitgelieferte Software beinhaltet einen *Kontrollteil* zur Kalibrierung und Beobachtung der Biosignale sowie einen *Konfigurationsteil* für die Profilerstellung.

[102] siehe [OCZ 2008, S. 11]

Nach der *Hardwareinstallation* des NIAs im *Microsoft Windows*[103]-Betriebssystem kann die NIA-Software gestartet werden.

Kalibrierungserfordernis

Zur Verbesserung der Signalverarbeitung wird von OCZ empfohlen, vor jeder Verwendung des NIAs eine Kalibrierung vorzunehmen. Sie bewirkt die Einstellung *„der Biosignale auf das optimale Signal-Rausch-Verhältnis“*[104]. Um einen erfolgreichen Abschluss zu erreichen, muss sich das gelbe Muskelsignal während der Kalibrierung unterhalb der grünen Linie befinden (siehe Abbildung 5.1). Der Vorgang selbst dauert zwar nur wenige Sekunden, es kann aber einige Minuten dauern bis sich ein zu hohes Muskelsignal senkt. OCZ spricht von der Zeit bis sich das *„optimale Elektrolyt-Gleichgewicht zwischen den Mikroporen der Sensoren und der Haut“*[105] einstellt.

Dauer?

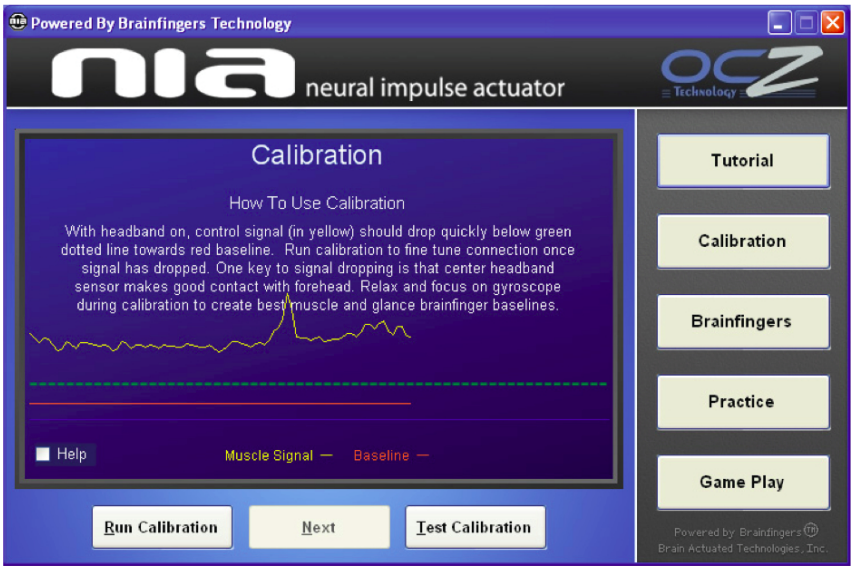

Abbildung 5.1: Kalibrierung vor jeder Benutzung des NIAs [OCZ 2008, S. 5]

eingeschränktes Biofeedback

Weiters können in der NIA-Kontrollsoftware die Zustände der Steuerungskanäle betrachtet werden. Auch OCZ nennt sie – wie BAT beim Cyberlink – „Brainfingers“ (siehe Abbildung 5.2). Erkenn- und einstellbar sind Kanäle für Augenbewegungen („Glance“, EOG), Muskelbewegungen („Muscle“, EMG) und die EEG-Signale in den Alpha- und Beta-Frequenzbereichen. OCZ bietet damit *„eingeschränktes Biofeedback“* zur Übung der Entspannung (für den Zusammenhang der Frequenzbereiche mit der Entspannung siehe Abs. 2.2, S. 10ff).[106]

[103] siehe `http://www.microsoft.com/windows` (Stand Sept. 2009)
[104] siehe [OCZ 2008, S. 6]
[105] siehe [OCZ 2008, S. 5]
[106] siehe [OCZ 2008, S. 14]

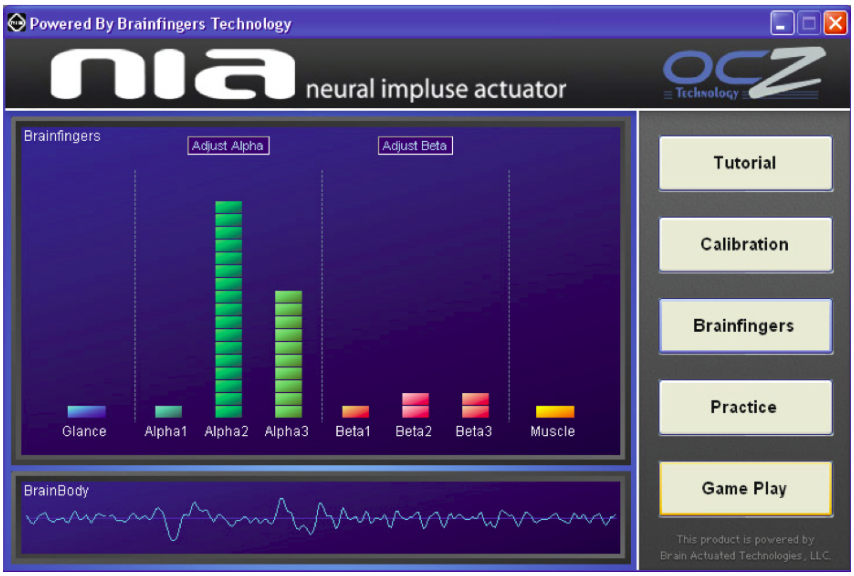

Abbildung 5.2: Anzeige der *Brainfingers* des NIAs [OCZ 2008, S. 14]

Das Cyberlink Brainfingers System bietet eine ähnliche[107] Brainfingers-Anzeige wie die NIA-Software (siehe Abbildung 5.3[108]). In [Bra 2004, S. 17] beschreibt BAT die Kanaldarstellung folgendermaßen: Brainfingers

Glance (F1-F3): Diese Balken steigen, wenn die Augen verstärkt nach links und rechts bewegt werden. EOG

Alpha (F4-F6): Eine Erhöhung kann erreicht werden, wenn sich die Rückenmuskulatur entspannt bzw. die Augen (bewusst) unfokussiert sind. BAT bezeichnet den Zustand ähnlich einer „Tagtraum"-Situation, wo das Gehirn *beruhigt* ist. EEG – α

Beta (F7-F10): Die Beta-Frequenzen treten bei „Aufregung" hervor, wodurch sich die Amplituden der entsprechenden Anzeige vergrößern. EEG – β

Muscle: Dieser Kanal reagiert auf Gesichtsmuskelkontraktionen, wie z. B. ausgelöst durch Augenbrauen-Heben oder Kieferbewegungen. Entspannung sollte den Balken senken. EMG

[107] Glance besteht beim Cyberlink aus drei Kanälen, im Gegensatz zu einem beim NIA
[108] Bild: `http://www.brainfingers.com/images/BasicScreen.jpg` (Stand Sept. 2009)

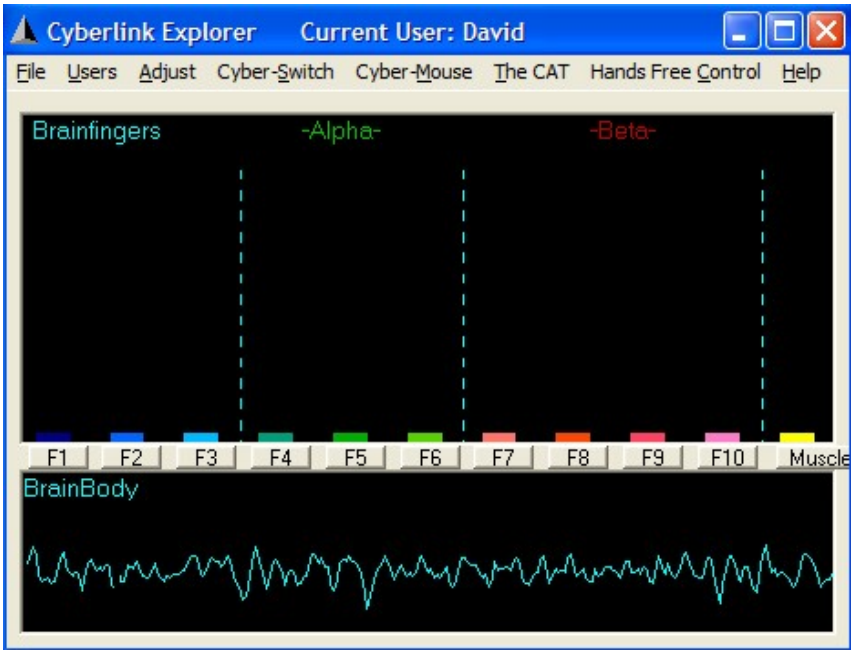

Abbildung 5.3: Die verfügbaren *Brainfingers* im Cyberlink-System von BAT

Lernspiel *Pong*

Zur Gewöhnung an den NIA beinhaltet die Kontrollsoftware ein einfaches Spiel namens *Pong*, welches über Brainfingers (vorwiegend Muskelsignale) zu steuern ist (siehe Abbildung 5.4). OCZ behauptet in diesem Zusammenhang die Steuerbarkeit von Pong innerhalb kürzester Zeit. Es können außerdem verschiedene Schwierigkeitsstufen gewählt werden.

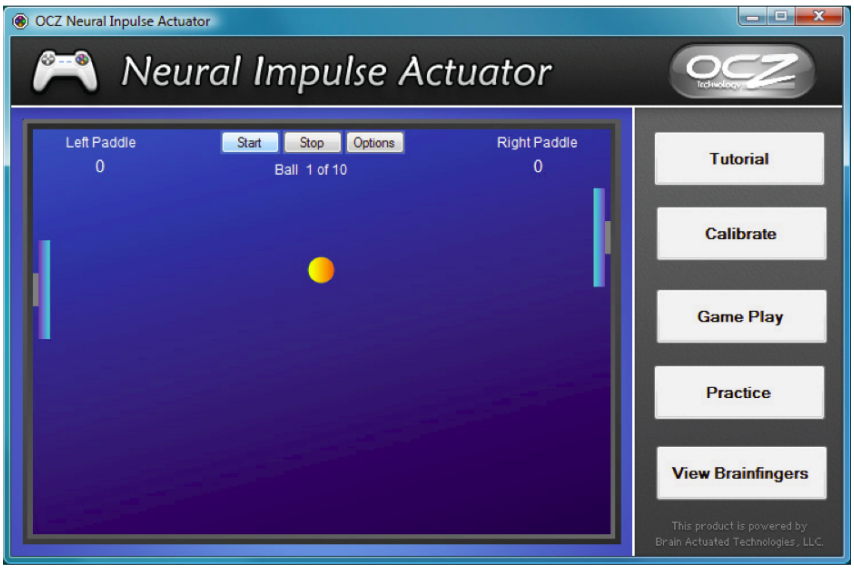

Abbildung 5.4: OCZ liefert mit *Pong* ein einfaches Lernspiel [OCZ 2008, S. 15]

evolutionäre Nutzung

Vielmehr baut das Konzept des NIAs aber auf „learning by doing“ mit den Lieblingsspielen der Benutzer. Der NIA soll dafür anfangs nur einfache Nebentätigkeiten[109] übernehmen, bis der Anwender die Kontrolle besser erlernt hat.

digitale Schalter

Um den NIA verwenden zu können, muss ein Profil gewählt werden, welches Brainfingers-Steuerkanäle auf Befehle zuweist. Einerseits ist es möglich drei[110] „Schalt“-Aktionen zu definieren (siehe Abbildung 5.5), andererseits können bis zu vier[111] „Joystick“-Aktionen verwendet werden. Schalter sollen dabei digitale Ereignisse wie z. B. Mausklicks oder Tastenanschläge mimen. Diese Schalter werden „betätigt“, sobald ein gewisser Schwellwert vom verwendeten Brainfinger überschritten wird.

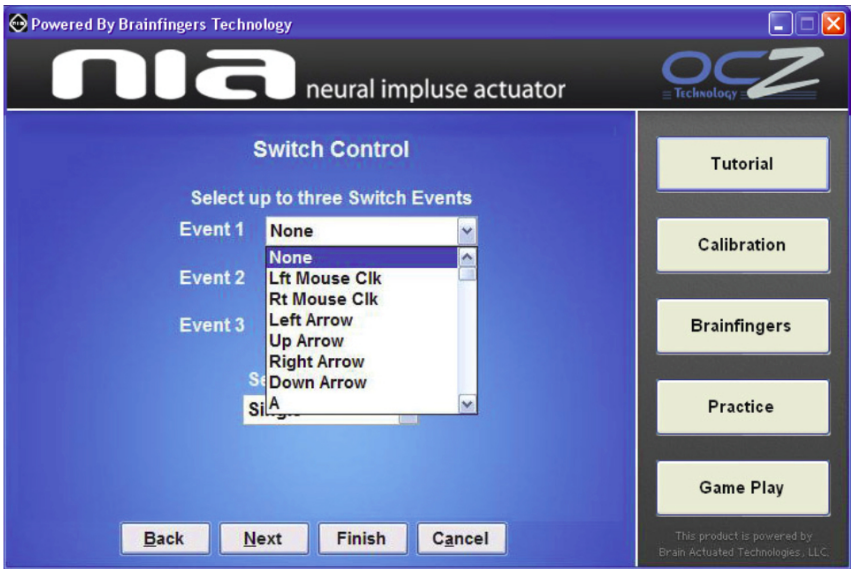

Abbildung 5.5: Die Konfiguration eines *Schalters* für *digitale Aktionen* [OCZ 2008, S. 18]

analoge Joysticks

Joysticks definieren für *Bereiche/Zonen* der Brainfinger-Signalwerte verschiedene Aktionen. Bringt man den Brainfinger über den ersten Schwellwert (in die erste Zone), so beginnt die Spielfigur z. B. vorwärts zu gehen. Steigt der Wert daraufhin in den folgenden Bereich, könnte eine Laufbewegung ausgelöst werden etc. OCZ empfiehlt die Definition von intuitiven Verhaltensweisen[112]. Bezugnehmend auf das eben formulierte Beispiel, ist eben Laufen eine anstrengendere Tätigkeit als Gehen und sollte daher in einen höheren, an das Gehen anschließenden, Bereich gelegt werden. Ein Joystick kann vier aufeinander folgende Ausführungszonen besitzen (Z1 bis Z4)[111].

4 Zonen

10 Wirkungsmodalitäten

Für die Aktionen der Joystick-Bereiche kann einer von zehn verschiedenen Wirkungs-

[109] z. B. Waffenwechsel in *Ego-Shooter*-Spielen
[110] siehe [OCZ 2008, S. 18]
[111] siehe [OCZ 2008, S. 19]
[112] siehe [OCZ 2008, S. 21]

modi gewählt werden (siehe Abbildung 5.6). Beispielsweise könnte der Eintritt in einen Bereich die Aktion nur einmal (Modus *Einfach* z. B. für Mausklick) oder öfters ausführen lassen (Modi *Verweilend...*, *Wiederholt...*, *Halten* z. B. für Halten oder intermittierendes Betätigen der Taste).[113]

Abbildung 5.6: Die Konfiguration eines *Joysticks* für *analoge/digitale Aktionen* [OCZ 2008, S. 19]

Parallel-joysticks

Gleichzeitigkeit

Joysticks können auch parallel definiert werden (siehe Abbildung 5.7). Das bedeutet die Verwendung von Brainfingers anderer Joysticks als zusätzliche Aktion in bestimmten Bereichen. Man erreicht damit gewissermaßen die Umsetzung zeitlich paralleler Ereignisse, wie sie Spieler z. B. mit gleichzeitigen Tastenanschlägen durchführen würden.[114]

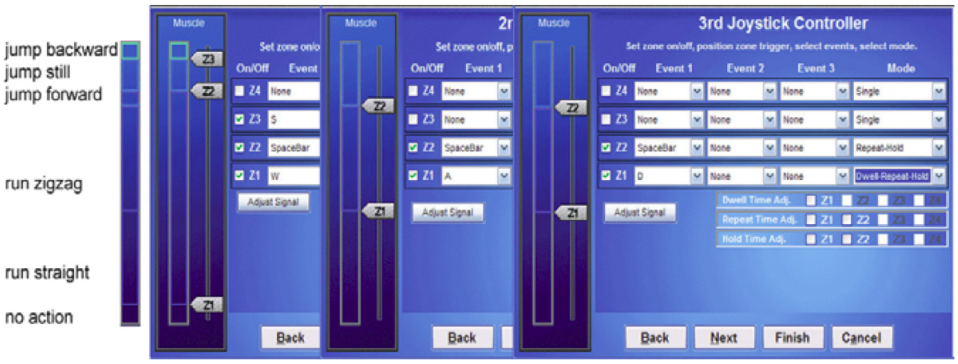

Abbildung 5.7: *Parallel-Joysticks* zur Simulation gleichzeitiger Aktionen (z. B. mehrfache Tastenanschläge) [OCZ 2008, S. 23]

[113] siehe [OCZ 2008, S. 20]
[114] siehe [OCZ 2008, S. 23]

flexible Einstellung

Die erfolgreiche Verwendung von Parallel-Joysticks bleibt gewiss dem gut geübten NIA-Benutzer vorbehalten. Es zeigt allerdings die mögliche Anwendungsflexibilität (siehe Abbildung 5.8[115]), was letztlich zu begrüßen ist. Für alle Brainfinger-Aktionen kann die *Verstärkung*, *Glättung* sowie *Schwellwerte* des Signals vom Benutzer angepasst werden[116].

flexible Verwendung

Komplexität?

Diese Anpassungsfähigkeit trägt abermals dazu bei, dass der NIA äußerst flexibel verwendbar ist. - Jedoch induzieren die gebotenen Möglichkeiten eventuell auch eine komplexe Bedienung. Es schmälert es die Lernmotivation, wenn wiederholt Einstellungen vorgenommen werden müssen/können, welche die Funktion verbessern oder aber auch verschlechtern. - Mit dem Ansatz von OCZ wird der Weg zur Beherrschung des NIAs dem Benutzer überlassen. Dies überfordert vor allem technisch weniger Interessierte.

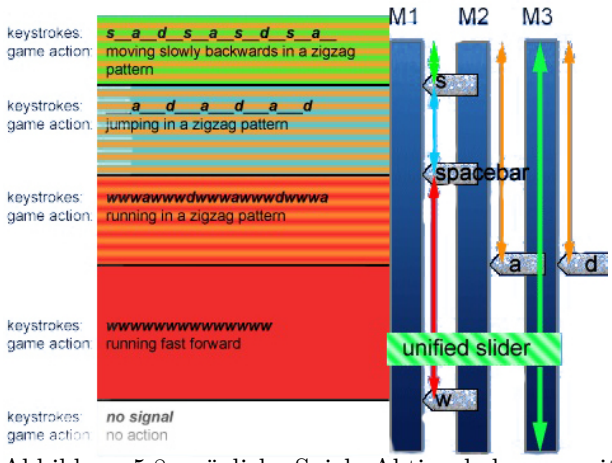

Abbildung 5.8: mögliche Spiele-Aktionsbelegung mit dreifach-*Parallel-Joysticks*

Motivation?

5.2 Die Software des Cyberlink Brainfingers Systems

Feedback-interesse

BAT versucht den Benutzer ihres Cyberlinks besser zu unterstützen. Sie liefern Lernsoftwareanwendungen, welche über Feedback das Erlangen der Kontrolle erleichtern. Es wird auch ein Dokument als Wegweiser für die erforderlichen Lernschritte bereitgestellt[117]. Der Leitfaden enthält fünf Schritte, welche es dem Benutzer ermöglichen sollen, den Cyberlink nach und nach besser steuern zu können. Im Wesentlichen beschreibt das Dokument, welche Cyberlink-Softwareanwendung wann und wie zum Training verwendet werden soll.

vielseitige Lernsoftware

Die Cyberlink-Software enthält demnach Anwendungen zum Erlernen von Klick-, (Maus)Bewegungs- und Steuerungsaufgaben. Unter anderem existiert eine Tastatur-Anwendung, das *Click-Game*, *Brain Billard Game*, *Grow Game*, *Pong Game*, *The*

[115] Bild: `http://www.lostcircuits.com/advice/ocz_nia/niaall2.gif` (Stand Sept. 2009)
[116] siehe [OCZ 2008, S. 26]
[117] siehe [Marler et al. 2006], abrufbar unter Datei: `http://www.brainfingers.com/Tutorial.pdf` (Stand Sept. 2009)

Maze Game oder das *Tetris Game*. Auch eine vollständig freihändige Bedienung der Cyberlink-Hauptanwendung über den Cyberlink kann durchgeführt werden.[118]

Kosten / Nutzen

Es ist bedauerlich, dass der NIA diese umfangreiche Trainingssoftware nicht enthält. Mit ihr wäre ein zielgerichtetes Erlernen der Steuerung wahrscheinlich einfacher. Andererseits sind die einschränkenden Maßnahmen verständlich, da der Cyberlink um ein Vielfaches (ca. 20 x[119]) mehr kostet.

5.3 Selbst erstellte NIA-Software

Verbreitung des NIAs

Der NIA hat, dank seines günstigen Preises und der neuartigen[120] Mensch-Maschine-Interaktion, rasch eine weite Verbreitung in der Spieleindustrie gefunden. Die fehlenden offiziellen Detailinformationen zu Hard- und Software (Programmierschnittstelle/SDK) haben einige Benutzer veranlasst, selbst Analysen und Entwicklungen anzustellen.

Ein Kreis unterschiedlichster NIA-Besitzer führt darüber im Internetforum der Firma OCZ unter `http://www.ocztechnologyforum.com/forum/forumdisplay.php?f=173`[121] eine angeregte Diskussion.

5.3.1 Frei verfügbare NIA-Software

vielfältige Verwendung

Einige der privat erstellten Programme sind:

NiaSharpReader – Ein C#-Programm, welches die NIA-Daten an der USB-Schnittstelle ausliest und diese in einem Liniendiagramm darstellt.
`http://nia-brew.googlecode.com/files/nia_ReaderV1.6.1.zip` (Stand Sept. 2009)
`http://www.ocztechnologyforum.com/forum/showthread.php?t=43298` (Stand Sept. 2009)

bearclaw – Ein C++-Programm, welches die NIA-Daten mittels Fourier-Analyse aufspaltet und darstellt.
`http://www.bearclaw.info/niaqwt.cc` (Stand Sept. 2009)

Linux Treiber/Software – Ein OpenSource-Projekt zur Verwendung des NIAs unter Linux.
`http://sourceforge.net/projects/nia4linux` (Stand Sept. 2009)

[118] siehe [Bra 2004] bzw. `http://www.brainfingers.com/cyberlink.htm` (Stand Sept. 2009)
[119] siehe Abs. 3.3, S. 16 bzw. Abs. 4.1, S. 20
[120] neuartig zumindest im Massenmarkt
[121] Stand Sept. 2009

EntrainerEEG – Erste Versuchssoftware in Java.
`http://entrainer.sourceforge.net` (Stand Sept. 2009)
`http://www.ocztechnologyforum.com/forum/showthread.php?t=59647` (Stand Sept. 2009)

PyNIA – Mac/Linux SDK in Python. Liest den NIA über die USB-Schnittstelle aus und führt diverse Signalkorrekturen durch.
`http://code.google.com/p/pynia/downloads/list` (Stand Sept. 2009)
`http://www.ocztechnologyforum.com/forum/showthread.php?t=54620` (Stand Sept. 2009)

TweetDreams – Baut auf PyNIA auf und versucht, während der Benutzer schläft, die REM-Schlafphase (*REM*: siehe Abs. 2.2, S. 11) zu erkennen. Die Integration des Twitter-Dienstes[122] ermöglicht die automatische Versendung von Nachrichten hinsichtlich des vorherrschenden Schlafzustandes.
`http://www.ngen.us/?p=21(StandSept.2009)`
`http://www.ocztechnologyforum.com/forum/showthread.php?t=58190` (Stand Sept. 2009)

NIA-Tortilla – Baut auch auf PyNIA auf und implementiert (in Python) eine Vielzahl an Signalverarbeitungsmöglichkeiten mit dem Ziel, diverse Brainfingers den gewünschten Aktionen zuweisen zu können. Es wird dabei eine „unbewusste“ Trainingsstrategie verfolgt, d. h. die Software lernt die Aktion-Brainfinger-Kombination (Klassifikation) während der Benutzer sie ausführt. Die Software lernt also wie das Gehirn des Anwenders arbeitet, nicht umgekehrt.
`http://www.ocztechnologyforum.com/forum/showthread.php?t=56238` (Stand Sept. 2009)

NIA Tones – Gibt Töne aus, wenn bestimmte Tasten gedrückt werden. Die Idee ist, dass das Gehirn lernt, die Töne mit den Aktionen in Verbindung zu bringen. Dadurch unterschiedlich auftretende Gehirnaktivitäten sollen eine Differenzierung der NIA-Signale erleichtern.
`http://www.ocztechnologyforum.com/forum/showthread.php?t=59188` (Stand Sept. 2009)

BrainRatX – Ein Script, welches die Mausaktionen auf sechs Tasten zuweist. Mit einem in der NIA-Software definierten Profil kann dadurch die Maus gesteuern werden.
`http://www.ocztechnologyforum.com/forum/showthread.php?t=51087` (Stand Sept. 2009)

[122] `http://twitter.com` (Stand Sept. 2009)

NIA Output Reader – Ermöglicht die Verwendung der NIA-Signale in EEG-Softwareanwendungen, welche als Client von NeuroServer[123] betrieben werden können (wie z. B. BrainBay[124]). Die NIA-Daten werden dafür über TCP/IP an den NeuroServer übertragen. BrainBay und NeuroServer sind Teile des OpenEEG-Projekts[125]. Weitere EEG-Anwendungen in diesem Projekt sind unter `http://openeeg.sourceforge.net/doc/sw` (Stand Sept. 2009) zu finden.
`http://www.ocztechnologyforum.com/forum/showthread.php?t=43489` (Stand Sept. 2009)

The Soundstone Project – Gibt die NIA-Signale in Tönen aus.
`http://www.ocztechnologyforum.com/forum/showpost.php?p=343643&postcount=11` (Stand Sept. 2009)

CenterView – Mauszeigeremulation für den NIA.
`http://www.ocztechnologyforum.com/forum/showthread.php?t=42370` (Stand Sept. 2009)

NIA / Lucid Dream Project – Ein Projekt, welches auch die REM-Schlafphase erkennt und den Benutzer dann in den Zustand eines *luziden Traumes* (oder *Klartraumes*) verhelfen soll. Interessant bei diesem Zustand ist, dass der Schlafende den Traum bewusst miterlebt.
`http://code.google.com/p/nia-remdreamer` (Stand Sept. 2009)

All diese Programme setzen das erarbeitete Wissen der NIA-Benutzer in vielseitiger Weise um. Besonders aufschlussreich in Bezug auf die Datengewinnung und -bearbeitung sind der *NiaSharpReader* und *PyNIA* (für Datengewinnung/USB) sowie *NIA-Tortilla* (für Signalverarbeitung).

5.3.2 Das Datenformat des NIAs

Eine der wichtigsten Erkenntnisse betrifft das Datenformat der NIA-Ausgaben. Unter `http://code.google.com/p/nia-brew/wiki/TechnicalSpecs`[126], sowie im offiziellen NIA-Forum[127] kann eine Zusammenfassung des bisher Bekannten gelesen werden:

Der NIA sendet seine Daten über USB nach der Spezifikation für *Human Device Interfaces* NIA⇝HID-Gerät

[123] `http://openeeg.sourceforge.net/doc/sw/NeuroServer` (Stand Sept. 2009)
[124] `http://www.shifz.org/brainbay` (Stand Sept. 2009)
[125] `http://openeeg.sourceforge.net/doc` (Stand Sept. 2009)
[126] Stand Sept. 2009
[127] siehe `http://www.ocztechnologyforum.com/forum/showthread.php?t=50776` (Stand Sept. 2009)

(HID)[128], also wie USB-Standardeingabegeräte. Das ist dahingehend wichtig, weil der Betrieb des NIAs deshalb i. A. keine eigene Treibersoftware erfordert. Das *Windows*-Betriebssystem[129] von *Microsoft* enthält bereits standardmäßig HID-Treiber[130]. Auch für andere Betriebssysteme existieren Wege die USB-Schnittstelle auszulesen bzw. gibt es HID-Treiber.[131]

Standardtreiber

Das HID-Protokoll und der NIA

Jedes HID-Gerät kennt eine *Report*-Datenstruktur für *Input*, *Output* und *Feature*. Die über den HID-Treiber gelieferten/zu sendenden Größen sind folgend:

Standard-protokoll

Input-Report	56 Bytes
Output-Report	9 Bytes
Feature-Report	2 Bytes
Biopotential-Datenformat	24-Bit-Integer (-8388608 bis 8388607)

Die Input-Pakete, welche der NIA sendet sind 56 Bytes lang und im *Hexadezimal*-Zahlenformat. Sobald der NIA mit der USB-Schnittstelle verbunden ist, sendet er 1024[132] Pakete pro Sekunde. Alle Pakete beginnen mit einem Byte `00` („ReportID 0“). Danach liegen 16 24-Bit-Ganzzahlen (*Integer*), welche die Biopotentialdaten darstellen. Die Bytereihenfolge entspricht dem *Little Endian*-Format, also beginnen diese Byte-Tripel jeweils mit dem niederwertigsten Byte des Integers. Die 16 freien Datenplätze im HID-Paket dienen als Buffer. Nicht verwendete Bereiche sind mit `00 12 7A` ($\hat{=}$ 0x`7A 12 00` = 0d 8 000 000) initialisiert.

1024 HID-Pakete / Sekunde

max. 16 Datenpunkte

Nach den NIA-Daten folgen zwei statische Bytes `38 BD`, sowie zwei weitere zwischen `FC FF` und `FE FF` schwankende Bytes. Die vorletzten zwei Bytes stellen einen *Zeitstempel* (=16-Bit) für das Paket dar. Der Zeitwert ist eine Zahl für den letzten Datenwert im Paket. Er ist mindestens um die Anzahl der enthaltenen Datenwerte höher als der Zeitstempel des zuletzt abgefragten HID-Pakets. Sollte der „Zeitabstand“ zwischen zwei Paketen größer sein, so sind Daten verlorengegangen, weil die letzte Abfrage

Zeitstempel

Bufferfähigkeit

[128] siehe `http://www.usb.org/developers/hidpage` (Stand Sept. 2009)
[129] siehe `http://www.microsoft.com/windows` (Stand Sept. 2009)
[130] siehe `http://msdn.microsoft.com/en-us/library/dd446410.aspx` (Stand Sept. 2009)
[131] siehe `http://www.libusb.org` (Stand Sept. 2009) für *Linux*, *FreeBSD*, *NetBSD*, *OpenBSD*, *Darwin*, *MacOS X*, *Windows*;
speziell für das OpenSource-Betriebssystem *Linux*: `http://www.linux-usb.org` (Stand Sept. 2009)
[132] siehe `http://www.ocztechnologyforum.com/forum/showthread.php?p=313645` (Stand Sept. 2009)

länger her war, als Bufferstellen vorhanden sind. Die letzte Stelle enthält die Anzahl der im Paket enthaltenen 3-Byte-Biopotentialdaten (=„nicht-`00 12 7A`“-Werte).

Beispielhaft soll nachfolgend ein 56 Byte HID-Inputpaket aufgegliedert werden:

HID-Struktur

B0	B1–B48		B49–B52		B53–B54	B55
00	88 05 84 87 05 84 AF 03 84	00 12 7A ··· 00 12 7A	38 BD	FD FF	69 0C	03
ID	3 mal Biopotentialdaten	13 mal Leerdaten	„Konstanten“		Zeitwert	#

Mit dem Wissen um das Datenformat ist es nun denkbar, die Möglichkeiten des NIAs in eigenen Softwareanwendungen auszunutzen. Im Bereich der Signalverarbeitung versierte Entwickler werden damit auch ohne eine offizielle Programmierschnittstelle innovative Ideen umsetzen können.

eigene Software möglich

5.3.3 NiaBlinkReader

Nach anfänglichen Versuchen des Autors mit der NIA-Original-Software konnten keine befriedigenden Ergebnisse erzielt werden; u. a. wegen hoher Prozessorauslastung, Kalibrierungsproblemen, scheinbarer Willkür der Brainfingers, Übung mit Spielen etc.

Wegen dieser Probleme mit der originalen NIA-Software muss der NIA, bezüglich der zu untersuchenden *einfachen, allgemeinen Computersteuerung*, als wenig tauglich eingestuft werden. Die Spielezentrierung sowie Lernmethodik ist für nicht-Spieleanwendungen unbefriedigend. *Wegen der Ableitung des NIAs vom Cyberlink Brainfingers System, dürfte die Hardware allerdings durchaus die notwendigen Eigenschaften aufweisen.*

Durch dieses Annahme wurde die Entwicklung einer eigenen NIA-Steuersoftware angedacht. Dazu mussten Informationen zur Funktionsweise des NIAs gesammelt und Applikationen anderer NIA-Benutzer (siehe Abs. 5.3.1, S. 38) studiert werden. Resultat war, dass sich die meisten der Anwendungen auf eine Gewinnung der Frequenzanteile[133] in den NIA-Daten konzentrieren. Frequenzanalyse war also der vorläufig erste Anknüpfungspunkt.

Eigen-entwicklung

5.3.3.1 Die Idee – einfaches „Blinzeln“

Bei Verwendung des *NiaSharpReaders*, welcher ausschließlich[134] die Rohdaten der

Blinzelmuster

[133] vorwiegend Alpha- und Beta-Wellen

USB-Schnittstelle in einem Diagramm anzeigt (siehe Abbildung 5.9), fiel auf, dass einfaches *Links-* bzw. *Rechts-Blinzeln* mit den Augenlidern sehr charakteristische Merkmale in den Potentialdaten erzeugt.

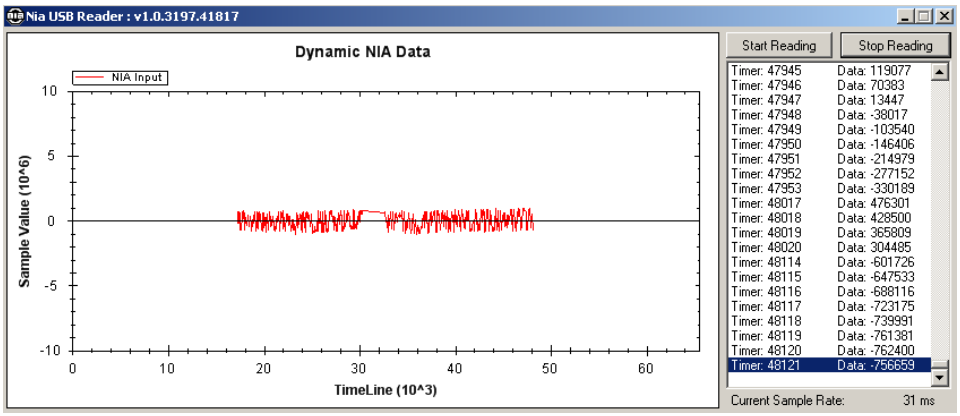

Abbildung 5.9: Das NiaSharpReader-Programm

charakteristische Unterschiede

Artefakte

Die Aktionen bewirken gegenläufige Muster, wobei je nach Auge ein *starker Signalausschlag* nach oben oder unten zu beobachten war; danach folgt zumeist ein leichtes „Ausschwingen“ auf die gegenüberliegende Seite (siehe Abbildung 5.10). Wie bereits bekannt, sind diese Ausschläge auf EOG[135]- und eventuell EMG[136]-Artefakte zurückzuführen (*Artefakte*: siehe Abs. 3.1, S. 13).

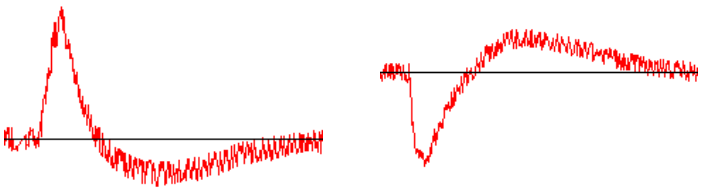

Abbildung 5.10: Charakteristische NIA-Potentialdaten-Muster für einmaliges „Blinzeln“: *linkes Auge* (links) bzw. *rechtes Auge* (rechts)

natürliches Blinzeln

Common Mode-Rejection

Ein weiterer Vorteil dieser Idee ist, dass das natürlich stattfindende zweiäugige Blinzeln keine besonderen Ausschläge erzeugt. Auch bewusstes, zweiäugiges Blinzeln unterschiedlicher Intensität folgt dieser Beobachtung. Weiß man um die *Common-Mode Rejection*-Anwendung (siehe Abs. 4.3, S. 29) in der NIA Hardware, lässt sich dieses Verhalten auch technisch begründen.

Menüsteuerung

Im Hinblick auf die Aufgabenstellung dieser Arbeit – „Steuerung von einfachen

[134] abgesehen von einer Verschiebung der Daten Richtung Nulllinie
[135] Augenbewegung
[136] Kopfbewegung, „Stirnrunzeln“ – *Frontalis*-Muskel

Menüsystemen mit dem NIA“ – erschien die Potentialdatenverarbeitung des Blinzelmusters als gangbarer Weg. Einfache Menüsysteme können nämlich mit wenigen verschiedenen Aktionen gesteuert werden und – viel wichtiger – sollte Augenblinzeln dem durchschnittlichen Menschen keine lange Trainingszeit abverlangen. Die Demotivation, ausgelöst durch eine schwierige Lernphase, wird somit verhindert.

einseitiges Blinzeln

Aufbauend auf dem *NiaSharpReader* (ca. 202 Zeilen ausführbarer Quellcode [LOC]) wurde der *NiaBlinkReader* in C# implementiert (siehe Abbildung 5.11). Der NiaBlinkReader ist eine experimentelle Software mit mehr als 10100 LOC, wovon ca. 3100 LOC vom Autor erdacht wurden und der Rest auf eingebunden Bibliothekscode entfällt. Die Software ist evolutionär entstanden; d. h. einige Signalverarbeitungsmethoden wurden ausprobiert und später teilweise wieder verworfen.

NiaSharpReader
⇓
NiaBlinkReader

experimentell

Der NiaBlinkReader erweitert den NiaSharpReader um – in den beteiligten Wissensgebieten – gebräuchliche Methoden der Signalaufbereitung sowie -verarbeitung. Dem experimentellen Charakter folgend, kann im Programm eine Vielzahl von Einstellungen vorgenommen werden können. Die Standardwerte entsprechen dabei den – vom Autor subjektiv – als passend erachteten Werten. Wie sich später noch herausstellen wird, müssen, abhängig von Umgebungsbedingungen sowie Benutzern, zumeist allerdings Adaptierungen vorgenommen werden.

eigene Wissensgebiete

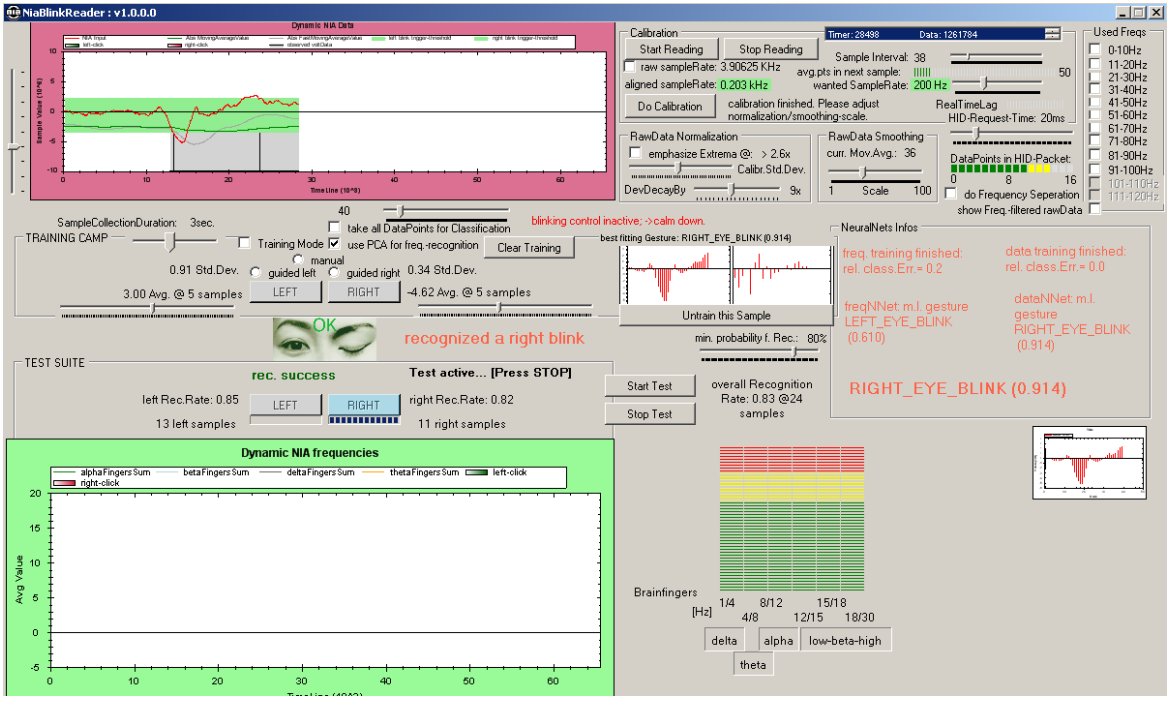

Abbildung 5.11: Das NiaBlinkReader-Programm mit einem NIA-„Ruhe“-Signalverlauf

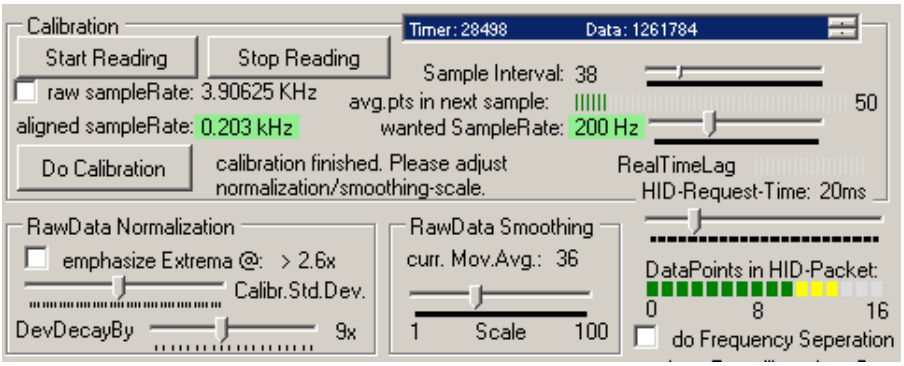

Abbildung 5.12: Detail – „generelle Einstellungen“

5.3.3.2 Einlesen der USB-Daten

Anm.: Die folgenden Verweise auf, in der Programmoberfläche zu findende, ≪Funktionen≫ können in Abbildung 5.11 bzw. Abbildung 5.12 (detaillierter) betrachtet werden.

Datenakquisition

Abrufintervall

Wurde die Auslesefunktion des NiaBlinkReaders (über ≪*Start Reading*≫) gestartet, so wird `ReadFromNiaSync()` (siehe Quelltext 5.1, [übernommene Funktionalität des NiaSharpReaders]) in einem 10 ms-Intervall aufgerufen. Diese HID-Request-Intervallzeit kann während der Ausführung (unter ≪*HID-Request-Time*≫) verändert werden.

Quelltext 5.1: `ReadFromNiaSync()`: Auslesen des NIAs über USB-HID-Treiber.

```
1   private void ReadFromNiaSync()
2   {
3       Byte[] inputReportBuffer = new Byte[56];
4       Boolean success = false;
5       ...
6           GenericHid.Hid.InputReportViaInterruptTransfer myInputReport = new GenericHid.Hid.
                InputReportViaInterruptTransfer();
7           myInputReport.Read(hidHandle, readHandle, writeHandle, ref NiaDetected, ref
                inputReportBuffer, ref success);
8
9           if (success)
10              Interpret(inputReportBuffer);
11      ...
12  }
```

Verzögerungen

Zu beachten ist allerdings, dass die *Timer*-Klassen in C# keine Präzision-Zeitgeber sind und der Ausführungszeitpunkt auch von der Prozessorauslastung abhängt. Im Programm existiert neben ≪*RealTimeLag*≫ ein ungefährer Indikator, wie weit die *realen* Abrufintervallzeiten hinter den eingestellten „nachlaufen“. Zumal auch die

beschränkten Bufferfähigkeiten[137] des NIAs in Betracht zu ziehen sind, kann ohne Vorkehrungen nicht von konstanten Sampling-Raten ausgegangen werden.

In `Interpret(Byte[] data)` wird dann die Behandlung der USB-Rohdaten begonnen. Zuerst müssen die Anzahl der Daten im Paket, der Zeitstempel und schließlich die Daten, wie in Abs. 5.3.2 beschrieben, extrahiert werden (siehe Quelltext 5.2).

Quelltext 5.2: Die Extraktion der Daten aus den HID-Paketen (in `Interpret(Byte[] data)`)

```
int validPackets = data[55];
...
ulong packetTimer = (ulong)(data[54] * (1 << 8)
             + data[53] * (1 << 0) - validPackets);//NIA's timer is 16bit
...

/*
* the 56B-array holds the values (24-Bit-integer) of the read-out nia samples.
* byte-56 says how many samples are within that byte[].
*
*/
for (ulong index = 0; index < (ulong)validPackets; index++)
{// go for all included/read samples

    ulong timerPosition = (ulong)(packetTimer + index);
   // construct one sample-value made of 3Bytes
   long rawData = data[index * 3 + 1] * (1 << 0) //*2^0
                            + data[index * 3 + 2] * (1 << 8) + //*2^8
                            data[index * 3 + 3] * (1 << 16);//*2^16
   ...
}
```

5.3.3.3 Korrektur der Sampling-Rate

Anm.: Die folgenden Verweise auf, in der Programmoberfläche zu findende, ≪Funktionen≫ können in Abbildung 5.11 bzw. Abbildung 5.12 (detaillierter) betrachtet werden.

Abschnittsbildung

Um das Problem der undefinierten Sampling-Rate zu lösen, wurde die *Zeitstempelinformation im HID-Paket* genutzt. Der Benutzer kann im Programm wählen, welche ≪*SampleRate*≫ (in Datenpunkten pro Sekunde; Hz) er für die Signalverarbeitungsschritte verwenden möchte. Nach dieser Information wird automatisch das passende ≪*SampleInterval*≫ gesucht, womit die angestrebte SampleRate zu erreichen ist. SampleInterval definiert dabei den zeitlichen Abstand, welchen zwei aufeinander folgende (endgültige) Datenpunkte aufweisen sollen.

[137] *Datenbuffer im HID-Paket*: siehe Abs. 5.3.2

Basis der Berechnung ist der angesprochene Zeitstempel. Alle Werte innerhalb eines Intervalls (die Anzahl ist im Programm neben ≪*points in next sample*≫ abzulesen) werden gesammelt, wobei deren Durchschnitt dem neuen Datum für dieses Intervall entspricht.

Basis: Zeitstempel

Durchschnittsbildung

Sollten, wegen zu langer NIA-Ausleseabstände[138], Intervalle keine Daten enthalten, so wird zwischen den zwei, diese „Leer-Intervalle“ einschließenden, Intervallen mit Wert *linear interpoliert*. Damit ist also die Sampling-Rate relativ gut „fixiert“. Kleine Schwankungen können sich durch unterschiedliche Prozessorauslastung ergeben. Die eingebaute *SampleInterval*-Automatik gleicht dies mit leichter Verzögerung aus.

Leerintervalle

Diese Zerteilung der Rohdaten in fixe Intervalle kann in `Interpret(Byte[] data)` nachvollzogen werden, diese Methode ist jedoch, nicht zuletzt durch den 16-Bit-Zeitstempel und dadurch auftretende *Overflows*, umfangreich. (siehe Quelltext A.1 im Anhang S. iii)

5.3.3.4 Datenaufbereitungsschritte

Nachdem nun die Datengewinnung abgeschlossen ist und ein zeitlich definierter (Intervall-)Punkt erzeugt wurde, startet der Signalverarbeitungsteil. Dafür wird in `Interpret(Byte[] data)` für jeden Punkt folgendes aufgerufen:

Signalverarbeitungsschritte

Quelltext 5.3: Die Signalaufbereitung für die Potentialdaten (in `Interpret(Byte[] data)`)

```
1 //process that point:
2       rawData = rawData >> 1; //max. 24b-int/2
3       rawData = RawDataNormalization(rawData);
4       rawData = RawDataPreProcessing((long)timerPosition, (long)rawData);
5       rawData = FrequencyAnalysis((long)timerPosition, rawData);
6       ...
7       UpdateZedGraph(timerPosition, rawData);
```

Normalisierung

Anm.: Die folgenden Verweise auf, in der Programmoberfläche zu findende, ≪Funktionen≫ können in Abbildung 5.11 bzw. Abbildung 5.12 (detaillierter) betrachtet werden.

Während der zu Beginn der Programmbenutzung durchzuführenden Kalibrierung (im Programm: ≪*Do Calibration*≫) werden Daten über das „Ruhesignal“ gesammelt.

Kalibrierung

[138] wegen eingestellter ≪*HID-Request-Time*≫ oder z. B. auch durch hohe Prozessorauslastung

Dies geschieht in `RawDataNormalization(long rawData)` und die Einzelvorgänge können in Quelltext 5.4 verfolgt werden.

Normalisierung

Nach der Kalibrierung wird das Signal mit den Ruhesignal-Informationen *mittelwertfrei* gemacht sowie die Standardabweichung korrigiert. Wie stark die Stauchung dabei ausfallen soll, kann vom Benutzer im Programm neben ≪*DevDecayBy*≫ bestimmt werden.

Extremwertbehandlung

Es besteht ebenso die Möglichkeit, Spitzenwerte im Signal verstärken zu lassen. Durch Auswahl von ≪*emphasize Extrema*≫ werden über einen (von der Kalibrierung abhängigen) Schwellwert liegende Signale verstärkt. Der Schwellwert ist im Programm unter ≪*emphasize Extrema*≫ regelbar. Der einzustellende Wert definiert einen Faktor, welcher als Produkt mit der Standardabweichung der Kalibrierungsdaten den Schwellwert bildet.

nichtlineare Verstärkung

Benutzerabhängigkeit

Die Verstärkung basiert auf der Potenzierung einer sigmoiden Funktion, um eine gewisse Nichtlinearität bzw. Begrenzung der Veränderung zu erreichen. ≪*emphasize Extrema*≫ lässt die Direktheit der Signalantwort in den Hintergrund treten und soll jene Benutzer unterstützen, welche Schwierigkeiten bei der Ausbildung der charakteristischen Blinzel-Amplituden haben. Wie vorteilhaft der Einsatz dieser Unterstützungsmaßnahme im Einzelfall ist, muss jedoch erst in Versuchen gezeigt werden. Standardmäßig ist die Funktion deaktiviert.

Quelltext 5.4: `RawDataNormalization(long rawData)`

```
1  private long RawDataNormalization(long rawData)
2  {
3      if (_inCalibrationMode)
4      {
5        _calibrationSamplesAvg = (_calibrationSamplesAvg *
6                                  _calibrationSamplesQueue.Count + rawData) /
7                                      (_calibrationSamplesQueue.Count + 1);
8        _calibrationSamplesQueue.Enqueue(rawData);//collect calibration samples
9       }
10      else if (_calibrationSamplesAvg > 0
11              && _normalizeOriginalData)
12      {
13         if (_calibrationSamplesStdDev.Equals(0))
14             _calibrationSamplesStdDev = CalculateStdDev(
15                        _calibrationSamplesQueue, _calibrationSamplesAvg);
16
17         //normalize rawVoltData
18        double rawDataDouble = (double)(rawData - _calibrationSamplesAvg);
19                                        // / _calibrationSamplesStdDev
20
21        double emphThreshold = _calibrationSamplesStdDev *
22                                       (double)trackBarEmphThres.Value / 10;
```

```
23        if (_emphasizeHighs && Math.Abs(rawDataDouble) > emphThreshold)
24            rawDataDouble = rawDataDouble *
25               Math.Pow(1+(1 / (1 + Math.Exp(-Math.Abs(rawDataDouble) / emphThreshold))),2);
26
27        if(_voltScaleFactor > 0)
28          rawData = (long)(rawDataDouble * 1024) >> _voltScaleFactor;
29        //else//std.dev=1 -> too low values to display
30        //  rawData = (long)(double)(rawDataDouble / _calibrationSamplesStdDev);
31      }
32      return rawData;
33    }
```

Glättung

Anm.: Die folgenden Verweise auf, in der Programmoberfläche zu findende, ≪Funktionen≫ können in Abbildung 5.11 bzw. Abbildung 5.12 (detaillierter) betrachtet werden.

gleitender Durchschnitt

Durch `RawDataPreProcessing(long timerPosition, long rawData)` (siehe Quelltext 5.5) wird eine Glättung der Daten in Form eines *einfachen gleitenden Durchschnitts* erreicht. Die Anzahl der einzubeziehenden Punkte kann im Programm unter ≪*RawData Smoothing*≫ gewählt werden. Die Erhöhung des gleitenden Durchschnitts kann vor allem in „verauschten“ Situationen, wo viele Störungen auf den NIA einwirken sehr wichtig werden. Die damit einhergehende Trägheitszunahme der Signale kann über Steigerung der ≪*Sample Rate*≫ ausgeglichen werden.

Quelltext 5.5: `RawDataPreProcessing(long timerPosition, long rawData)`

```
1       private long RawDataPreProcessing(long timerPosition, long rawData)
2        {
3          //raw data as moving average
4          if(_smoothingQueue ==null )
5            _smoothingQueue = new Queue<double>(_smoothingFactor);
6
7          double valueToRemove = 0;
8          while (_smoothingQueue.Count >= _smoothingFactor)
9          {
10           _smoothedRawDataSum -= _smoothingQueue.Dequeue();
11         }
12
13         _smoothingQueue.Enqueue(rawData);
14
15         _smoothedRawDataSum += rawData;
16         return (long)(double)(_smoothedRawDataSum / _smoothingQueue.Count);
17       }
```

letztendlich sauberes Signal

Durch all diese Bearbeitungsschritte entsteht ein „sauberes“ NIA-Signal. Zum Ver-

gleich sollte hiezu Abbildung 5.13 mit Abbildung 5.10 verglichen werden. Die roten (ausgeprägten) Linien entsprechen dabei dem Datensignal; grau und grün in Abbildung 5.13 sind gleitende Durchschnittslinien (100 bzw. 500 Werte), welche hier jedoch nicht relevant sind. Ein Blinzelmuster (inkl. „Ausschwingen“) dauert in etwa zwei bis drei Sekunden.

Abbildung 5.13: Charakteristische NIA-Potentialdaten-Muster für einmaliges „Blinzeln“ nach der Datenaufbesserung: *linkes Auge* (links) bzw. *rechtes Auge* (rechts)

5.3.3.5 Klassifikation

Idee: Mustererkennungsautomatik

Nach diesen unerwartet[139] vielversprechenden Ergebnissen, lag also die Vermutung nahe, dass die Gestenmuster auch vom Computer automatisch erkannt werden könnten. Wie die Mustererkennung im Programm umgesetzt ist, wird in diesem Abschnitt eingehend erklärt.

Dezidiert herausgegriffen und beschrieben werden dafür nur die wichtigsten Quelltextbereiche. Ein ausgedehnteres Abdrucken erscheint wegen möglicher Änderungen („experimentelle Software“) bzw. der Übersichtlichkeit, nicht sinnvoll zu sein; die Implementierungsdetails können auf mannigfaltige Weise umgesetzt werden. Erwähnenswert zur Klassifikationsthematik ist vielmehr das zugrundeliegende Prinzip sowie die individuellen Ideen.

Es folgt also eine informelle Beschreibung mit kurzen Codefragmenten. Auf bereits besprochene Erkenntnisse und Theorien wird bei deren programmtechnischer Anwendung hingewiesen. Zugrundeliegende Methoden der Signalverarbeitung werden zum besseren Verständnis an entsprechender Stelle erklärt; für weiterführende Informationen können die jeweils angegebenen Quellen konsultiert werden.

Klassifikationsprozedere

Anm.: Die folgenden Verweise auf, in der Programmoberfläche zu findende, ≪Funktionen≫ können in Abbildung 5.11 bzw. Abbildung 5.14 (detaillierter) betrachtet werden.

diskontinuierliche Erkennung

Die Klassifikationsmethoden arbeiten aus Effizienzgründen nicht kontinuierlich. Viel-

[139] wenn man z. B. die Signale in der NIA-Original-Software betrachtet

mehr muss die Klassifikation erst durch ein bestimmtes Ereignis gestartet werden. Die Auslösung erfordert die Erreichung eines Schwellwertes in den Potentialdaten. Es existieren jeweils verschiedene Schwellwerte für das Blinzeln des linken bzw. rechten Auges. Die Schwellwerte leiten sich aus einem Trainingsmodus ab, in welchem die Software vom Benutzer „lernt“, wie *seine* Blinzelmuster zusammengesetzt sind.

Schwellwert-abhängigkeit

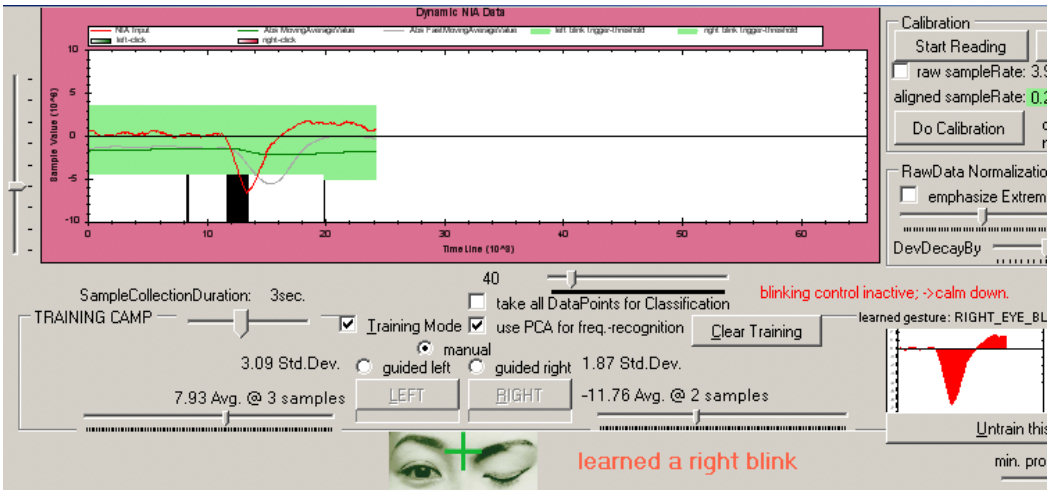

Abbildung 5.14: Detail – „Trainingsumgebung“

Das Training wird durch Auswahl in ≪*TRAINING CAMP*≫ neben ≪*Training Mode*≫ gestartet. Es besteht die Möglichkeit eines manuellen Trainings (≪*manual*≫) oder die geführte Absolvierung mit einer Automatik (≪*guided left*≫ bzw. ≪*guided right*≫). Im manuellen Modus löst die Betätigung von ≪*LEFT*≫ bzw. ≪*RIGHT*≫ die Signalaufzeichnug aus, woraufhin entsprechend einmalig geblinzelt werden soll.

Im geführten Modus wird die Aufzeichnung nach der vollständigen Füllung der Fortschrittsanzeige automatisch ausgelöst. Das darauffolgende Blinzeln mit dem gewählten (*guided left/right*) Auge soll erst danach beginnen. Im Trainingsprogramm sind *Beginn*, *Hochpunktsuche* und *Ende* der Aufzeichnung im Potentialdaten-Liniendiagramm mit vertikalen, schwarzen Strichen gekennzeichnet.

Feedback im Training

Das zuletzt aufgezeichnete Muster kann isoliert im *Gesture*-Diagramm betrachtet werden. Nicht saubere oder fehlgeschlagene, aber aufgezeichnete Aktionen, können über ≪*Untrain this Sample*≫ rückgängig gemacht werden. Diese fließen dann nicht mehr als Referenzbeispiele ein. Die Trainingsvorgänge werden im Bereich unterhalb von ≪*TRAINING CAMP*≫ und oberhalb von ≪*TEST SUITE*≫ visuell (mit Bildern und Statusmeldungen) unterstützt.

Die gesamte Geste sollte innerhalb des gekennzeichneten Bereichs abgeschlossen sein.

Anpassungen an Benutzer

Die im Programm unter ≪*SampleCollectionDuration*≫ einstellbare Zeit definiert, wie lange die Software das zu trainierende Muster aufzeichnet (die Zeitsteuerung übernimmt das `SampleCollectionTimer`-Objekt). Bei kürzeren oder längeren Gestenmustern kann dahingehend eine Anpassung vorgenommen werden. Abhängig von der eingestellten ≪*SampleRate*≫ ergibt sich somit eine gewisse Anzahl an Punkten für jedes Muster. Im Zuge eines Trainingsdatensatzes sollte die Dauer deshalb allerdings nicht verändert werden. Bei Bedarf können alle aufgenommen Trainingsbeispiele mittels ≪*Clear Training*≫ entfernt werden.

Trainingsstatistik

Durch das Blinzeln und die Bekanntgabe, welches Auge blinzelt, startet der `SampleCollectionTimer`. Im Wesentlichen wird während der Aufzeichnung der Hochpunkt im ausgeprägten Wellenberg lokalisiert (siehe Quelltext 5.6); das sich dort ergebende Vielfache zur Standardabweichung der Kalibrierungsdaten, ist Basis der späteren Statistik.

Quelltext 5.6: Finden des höchsten Punktes im Muster während der Aufzeichnung (in `RecognizeOnData(double time, double value)`)

```
1          double absValue = Math.Abs(value);
2          ...
3          // recognize blinking:
4         if (_blinkingAllowed &&
5             ( ...
6              (_inTrainingMode || _pendingTrainingSample))
7        {//triggered a possible blink
8          ...
9          if (_inTrainingMode)//will be false after capturing started
10         {//starting to find peak of this right-blink
11            if(_trainRight)
12              MarkSupposedClick(Gesture.RIGHT_EYE_BLINK);
13            else if(_trainLeft)
14              MarkSupposedClick(Gesture.LEFT_EYE_BLINK);
15
16           _highPointValue = value;
17           _pendingTrainingSample = true;//waiting to find max
18         }
19         else if (_pendingTrainingSample && _collectTrainSamples)
20         {
21            if ((absValue / _highPointValue) > 1)
22           {// found a new peak
23             if (_trainRight)
24               MarkSupposedClick(Gesture.RIGHT_EYE_BLINK);
25             else if (_trainLeft)
26               MarkSupposedClick(Gesture.LEFT_EYE_BLINK);
27
28             _highPointValue = absValue;
29           }
30         }
31         ...
32         _inTrainingMode = false;
33       }
```

Nach Ablauf des Zeitgebers werden die Daten des zugehörigen Musters ausgewertet (siehe Quelltext 5.7 sowie Quelltext 5.8). Der Durchschnitt der Vielfachen aus mehreren Trainingsbeispielen lässt eine gewisse natürliche Intensität des Blinzelvorganges eines jeden Benutzers erkennen. Weiters ist ersichtlich auf welcher Seite (positive oder negative Potentialdaten) die charakteristische „Welle“ für ein Auge auftritt.

Quelltext 5.7: `SampleCollectionTimer_Tick(object sender, EventArgs e)`: Analyse und Speicherung einer Trainingsgeste nach der Aufzeichnung

```
1   private void SampleCollectionTimer_Tick(object sender, EventArgs e)
2   {
3       //only in training-mode
4       if (chkTrainMode.Checked)
5       {
6           _collectTrainSamples = false;
7           TrainedGesturePeakReached();
8           if (_dataSamplesToClassifyQueue != null)
9           {
10              ...
11              AddDataTrainingSample(_dataSamplesToClassifyQueue, _gestureToCollect);
12          }
13          ...
14          SampleCollectionTimer.Stop();
15      }
16  }
```

Quelltext 5.8: `TrainedGesturePeakReached()`: Hochpunktauswertung

```
1   private double TrainedGesturePeakReached()
2   {
3       // peak reached. update the threshold
4       _pendingTrainingSample = false; // we finished training this blink
5       double trainedMultiplier = _newMulitplier / _calibrationSamplesStdDev;
6       Image bitmap = null;
7       if (_trainRight)
8       {
9           lblRight.Text = LEARNED_RIGHT;
10          bitmap = Bitmap.FromFile(".\\rightBlinkLearned.jpg");
11
12          MarkSupposedClick(Gesture.RIGHT_EYE_BLINK);
13          _rightTrainingSamples.Add(trainedMultiplier);
14          CalculateTrainingStatistics(Gesture.RIGHT_EYE_BLINK);
15      }
16      else if (_trainLeft)
17      {
18          lblLeft.Text = LEARNED_LEFT;
19          bitmap = Bitmap.FromFile(".\\leftBlinkLearned.jpg");
20          MarkSupposedClick(Gesture.LEFT_EYE_BLINK);
21          _leftTrainingSamples.Add(trainedMultiplier);
22          CalculateTrainingStatistics(Gesture.LEFT_EYE_BLINK);
23      }
```

```
24            if (bitmap != null)
25                picFace.Image = bitmap;
26            return _highPointValue;
27        }
```

lernen vom Benutzer

Die Trainingsaufzeichnung wird durch Abwahl von ≪*Training Mode*≫ beendet. In der darauffolgenden Phase, wo das Programm inaktiv erscheint, „lernt“ die Software die Trainingsbeispiele. Die Wartezeit (von einigen Sekunden bis Minuten – je nach Anzahl der zu erlernenden Muster) sollte den Benutzter daher nicht verwundern.

Feedback zur Erkennung

Nach der Trainingsphase – also in der Erkennungsphase – löst ein Überschreiten des 85 %-Durchschnittswertes[140] der Trainings-Hochpunkte die Gesten-Klassifikation aus (siehe Quelltext 5.9; die Schwellen sind im Potentialdaten-Liniendiagramm als Begrenzungen der grünen Schattierung, ober- und unterhalb der Nullinie zu erkennen). Dabei wird ein fortschreitender Bereich auf das Blinzelmuster überprüft. Die Länge dieses, im Potentialdaten-Liniendiagramm des Programms *grau* dargestellten, Bereiches bestimmt sich aus der Zahl der, in den Trainingsbeispielen, beobachteten Punkte. Der Anfang und das Ende eines Klassifikationsversuchs wird durch vertikale, schwarze Striche erkenntlich.

Quelltext 5.9: Beginn der Erkennung durch Schwellwertübertretung (in `RecognizeOnData(double time, double value)`)

```
1         double absValue = Math.Abs(value);
2         double rightThreshold = 0.85 * _calibrationSamplesStdDev *
              _rightBlinkAvgThresholdMulitplier;
3         double leftThreshold = 0.85 *_calibrationSamplesStdDev *
              _leftBlinkAvgThresholdMultiplier;
4
5         bool possibleRight = false;
6         bool possibleLeft = false;
7          // recognize blinking:
8         if (_blinkingAllowed &&
9             (!chkTrainMode.Checked &&
10             Math.Abs(rightThreshold) > 0 && Math.Abs(leftThreshold) > 0 &&
11             ((possibleRight = ((value / rightThreshold) > 1)) ||
12              (possibleLeft = ((value / leftThreshold) > 1)))) ||
13            (_inTrainingMode || _pendingTrainingSample))
14        {//triggered a possible blink
15          timer1.Stop();
16          ToggleBlinkingAbility(false);//freeze mode on
17
18          if (!_pendingTrainingSample && !chkTrainMode.Checked)
19          {
20            Image bitmap = null;
21            lblLeft.Text = string.Empty;
22            lblRight.Text = string.Empty;
23            if (possibleRight)
```

[140] Der automatisch errechnete Schwellwert kann im Programm bei Bedarf manuell geändert werden.

```
24              {
25                lblRight.Text = RECOGNIZED_POSSIBLE_BLINK;
26                _recognizedGesture = Gesture.RIGHT_EYE_BLINK;
27                MarkSupposedClick(Gesture.RIGHT_EYE_BLINK);
28                bitmap = Bitmap.FromFile(".\\rightBlinkPossible.jpg");
29              }
30              else if (possibleLeft)
31              {
32                lblLeft.Text = RECOGNIZED_POSSIBLE_BLINK;
33                _recognizedGesture = Gesture.LEFT_EYE_BLINK;
34                MarkSupposedClick(Gesture.LEFT_EYE_BLINK);
35                bitmap = Bitmap.FromFile(".\\leftBlinkPossible.jpg");
36              }
37              if(bitmap != null)
38                picFace.Image = bitmap;
39              _possibleGestureHappenedRecTimer = //nnet will classify it
40                (int)((double)(_recognDataSamplesCount));//*2)/3);//when trigger hits, we are
                      almost at peak
41              lblDataNetRec.Text = string.Empty; //now searching for new one
42            }
43            ...
44          }
```

Klassifikationsdauer

Die Erhöhung der Variable `_possibleGestureHappendRecTimer` in Quelltext 5.9 (Zeile 39) auf `_recognDataSamplesCount` ist für die Durchführung der Klassifikation entscheidend. Sie ist verantwortlich dafür, dass nach der Überschreitung des Schwellwertes nur jene Datenmenge überprüft wird, welche im Laufe des Trainings als Gestenlänge definiert wurde. Danach ist der ausgelöste Klassifikationsversuch beendet; die gewünschte Diskontinuität der Gestenerkennung ist damit erreicht.

Freeze Mode

Nach dem Versuch bleibt die Schwellwertüberwachung so lange deaktiviert, bis sich die gleitenden Durchschnittswerte 100 (grau) bzw. 500 (grün) nur noch um 30 % unterscheiden („Freeze Mode“). Die Inaktivität soll also eine Beruhigung des Signals erwirken und während dieser Zeit sind die Diagrammumrandungen rot eingefärbt. Sobald eine erneute Klassifikation möglich ist, weicht das Rot einem Grün; Schwellwertüberschreitungen lösen dann auch wieder eine neuerliche Gestenerkennung aus.

Künstliche Neuronale Netze

Für die Klassifikaktionsaufgaben wurden *Neuronale Netze* eingesetzt[141]. Eine andere Möglichkeit wäre z. B. der Einsatz von *Support Vector Machines (SVM)*[142]; SVM-

[141] durchaus üblich: siehe z. B. [Übeyli 2009], [Subasi 2006], [Jahankhani et al. 2006] oder [Pittner und Kamarthi 1999]
[142] siehe z. B. [Benimeli und Sharman 2007]

Algorithmen sind mächtiger als Neuronale Netze, doch konnten dahingehend keine freien Bibliotheken für C# gefunden werden. Weiters erscheint im Rahmen dieses „Proof-of-Concept"-Projektes eine eigene Implementierung von SVMs nicht sinnvoll.

Prinzipiell arbeiten Neuronale Netze ähnlich dem menschlichen Gehirn. Eine Vielzahl von sog. *Neuronen* sind untereinander vernetzt und beeinflussen sich gegenseitig mehr oder weniger. Neuronale Netze sind *fehlertolerant*, d. h. Schwankungen in den Eingabewerten werden toleriert, ohne die Klassifikation gänzlich versagen zu lassen. Daher sind Neuronale Netze in der Lage, eine *Generalisierung* zu schaffen; sie können also auch mit noch nicht bekannten Datensätzen umgehen und die Ähnlichkeit zu zuvor gelernten Referenzdaten bewerten.

Lernfähigkeit Neuronaler Netze

Die Netzstruktur[143] entsteht aus *Neuronen*, welche in *Input-*, *Hidden-* und *Output-Schichten* angeordnet sind. Zumeist ist jedes Neuron mit allen Neuronen einer anliegenden Schicht verbunden, spezielle Aufbauten sind aber möglich[144].

Vernetzung

Über *Gewichte* an den Verbindungen sowie *Aktivierungsfunktionen* in den Hidden- und Output-Neuronen breiten sich die Eingabewerte aus den Input-Neuronen im Netz aus. *Die Gewichtungen schwächen den Wert zum einen Nachfolger generell ab, können ihn jedoch auch verstärken.* Ein Hidden-Neuron verwendet üblicherweise nichtlineare Aktivierungsfunktionen[145] und kann dadurch wiederum nichtproportionale Änderungen für die Nachfolger bewirken. An den Output-Neuronen[146] erscheinen am Ende die Ergebnisse der „Berechnung". Damit sind *auch hochgradig nichtlineare Probleme mit Neuronalen Netzen numerisch lösbar.*

Informationsfluss

Die Analogie zum menschlichen Gehirn gründet auf den Gewichten und der Aktivierungsfunktion. Hohe Gewichte spiegeln starke[147] Synapsenverbindungen zwischen zwei Neuronen wider; Null-Gewichte bedeuten keine Abhängigkeit. Die Aktivierungsfunktion folgt dem Prinzip des *Aktionspotentials*[148] eines jeden Neurons. (*Gehirnfunktion*: siehe Abs. 2.1)

Gehirnähnlichkeit

In der Initialisierungsphase werden bekannte Beispieldaten im Neuronalen Netz „trainiert". Je mehr dabei trainiert wird, desto besser „lernt" das Netz wie Ein- und Ausgabe miteinander korreliert sind. Oftmaliges Training auf wenigen Trainingsdaten ist allerdings ab einem gewissen Zeitpunkt nachteilig, da sich das Netz

adäquate Lernphase

Overfitting-Vermeidung

[143] *Feed Forward Architektur*
[144] z. B. können gewisse Schichten über *Bias*-Neuronen beeinflusst werden
[145] z. B. *sigmoide* oder *Schwellwert*-Funktionen
[146] Output-Neuronen besitzen üblicherweise lineare Aktivierungsfunktionen
[147] kurze Synapsen, eng beieinander liegender Neuronen, schwächen das Potential z. B. weniger ab
[148] mögliche *Schwellwertgatter* als Aktivierungsfunktion entsprechen dabei dem Aktionspotential vollkommen; Sperrwirkung bis zum Schwellwert – darüber erfolgt die Wertweitergabe

vermehrt auf diese Daten einstellt („Auswendiglernen“, *Overfitting*). Die angestrebte Generalisierung für später zu klassifizierende Daten könnte dadurch verschlechtert werden. Je mehr unterschiedliche Trainingsdaten, desto besser die späteren Ausgaben; andererseits steigt damit auch der Trainingsaufwand.

Neuronale Netze können z. B. für Funktionsapproximation/Regression oder generell für Klassifikationsaufgaben eingesetzt werden. Im letzteren Fall sind die Werte an den Output-Neuronen als Wahrscheinlichkeiten für die Zugehörigkeit der Inputwerte zu einer bestimmten Klasse zu deuten.

Feature-Reduktion

Die Verwendung von relevanten Eingabedaten spielt eine wesentliche Rolle für die Klassifikationsleistung Neuronaler Netze. Das Netz stellt sich nämlich während des Trainings auf alle Inputwerte ein, wodurch insignifikante „Features“ die Kapazität des Netzes unnötig „verbrauchen“. Eingabewerte sollen daher soweit wie möglich reduziert werden, ohne zu viel Information zu beseitigen.

Nähere Informationen zu Neuronalen Netzen finden sich z. B. in [Lippe 2007], [Paetz 2006], [Weicker 2007] oder [Kluge 2006, 157ff].

Für die Implementierung der neuronalen Netze wurden Klassen der *ALGLIB*[149]-Bibliothek gewählt. Dies ist eine Algorithmensammlung aus dem Bereich der numerischen Datenverarbeitung und -analyse. Interessant ist, dass die Bibliothek in mehreren Programmiersprachen frei zur Verfügung steht.

Neural Net Ensembles

Parametertuning

Konkret wurden im NiaBlinkReader *early stopping Neural Network Ensembles*[150] zur Gesten-Erkennung eingesetzt. Dies entspricht einem *Klassifikationsproblem*, welches darin besteht, das betrachtete Muster einer bestimmten Geste zuzuweisen. Ensembles haben den Vorteil, dass die oft schwierige Auswahl guter Netz-Parameter entfällt. Ensembles enthalten nämlich mehrere Netze, welche ihrerseits jeweils sehr *redundant*[151] aufgebaut sein sollen. Die Gewichtsverteilung im finalen Netz leitet sich automatisch aus „Durchschnittsbetrachtungen“ an den Einzelnetzen ab.

Early Stopping

Sehr mächtige/redundante Neuronale Netze lernen durch ihre hohe Kapazität sehr genau und das Overfitting tritt schnell ein. *Early Stopping* bezeichnet eine Methode zur Vermeidung des Overfitting-Phänomens. Dafür werden die Trainingsdaten in Lern- und Validierungsdaten zerteilt. Das Lernen kann abgebrochen werden, wenn der Fehler auf den Validierungsdaten wieder zu steigen beginnt, weil dies ein Indiz dafür ist, dass sich die angestrebte Generalisierung zu verschlechtern beginnt.

[149] siehe `http://www.alglib.net` (Stand Sept. 2009)
[150] siehe `http://www.alglib.net/dataanalysis/mlpensembles.php` (Stand Sept. 2009)
[151] *redundant*: (zu) viele Neuronen für die Aufgabe $\rightsquigarrow$ (zu) hohe Lernkapazität

Ensembles haben somit eine Reihe von Vorteilen, welche den Umgang mit Neuronalen Netzen erleichtern. Nachteilig ist die längere Trainingszeit von Ensembles. *Early Stopping* benötigt durch die Datenteilung zudem mehr Trainingsbeispiele.

lange Trainingszeit

Potentialmusterklassifikation

Im Falle der NIA-Biopotentialdaten dienen diese als Eingabewerte in ein Neuronales Netz und linkes bzw. rechtes Blinzeln als mögliche Output-Klassen. Die Einführung der Datenteilung in konstante Intervalle (*fixe Sampling-Rate*: siehe Abs. 5.3.3.3) brachte eine signifikante Steigerung der Klassifizierungsgenauigkeit.

Um das Netz so gut wie möglich auf die Gesten einstellen zu können, werden die Mustertrainingsdaten folgendermaßen vorbereitet:

Featureaufbereitung

Im Anschluss an die Trainingsphase werden die ausgeprägten Wellenberge der Beispiele zentriert. Dies beseitigt Abhängigkeiten vom Blinzelzeitpunkt nach Auslösung der Signalaufzeichnung. Neuronale Netze haben eine fixe Anzahl an Input-Neuronen; die Idee für gute Klassifikationsleistungen ist nun, dass der Wellenberg im Blinzelmuster idealerweise immer von den gleichen Neuronen trainiert werden soll.

Musterzentrierung

Jene, durch die Verschiebung, an den Rändern auftretenden Fehlstellen werden mit Signaldaten gefüllt, welche *vor* der Aufzeichnung auftraten[152]. Dort sind im Allgemeinen nämlich relativ „ruhige“ Signale anzunehmen, weil der „Freeze Mode“ eine Korrelation von aufeinander folgenden Trainingsbeispielen verhindern soll. Weiters müsste die Intention des Benutzers saubere Trainingsmuster zu erhalten, ihn von „Störaktionen“ vor der Beispielaufzeichnung abhalten. Der maximale Abstand der jeweiligen Hochpunkte zu den Rändern definiert die Länge des zentrierten Musters (neue Länge = 2-facher max. Abstand) und wo – vorne oder hinten – demnach eine Fehlstelle auftritt. (siehe Quelltext 5.10)

Freeze Mode

Quelltext 5.10: Verschiebung des Hochpunktes im Muster in die Mitte (in `SampleCollectionTimer_Tick(object sender, EventArgs e)`)

```
1
2                   double foundHighPoint = Math.Abs(TrainedGesturePeakReached());
3                   ...
4                   double[] originalInputData = _dataSamplesToClassifyQueue.ToArray();
5                   double currAbsValue = 0;
6                   double maxAbsValue = 0;
7                   int maxAbsValueIdx = 0;
8                   for (int pointIdx = 0; pointIdx < originalInputData.Length; pointIdx++)
9                   {//find the maximum
```

[152] Ist der Buffer für diese Daten leer, wird mit Null ergänzt.

```
10                      currAbsValue = Math.Abs(originalInputData[pointIdx]);
11                  if (foundHighPoint == currAbsValue)
12                  {
13                      maxAbsValueIdx = pointIdx;
14                      maxAbsValue = foundHighPoint;
15                      break;
16                  }
17              }
18
19              int maxDistanceToBack = _dataSamplesToClassifyQueue.Count - maxAbsValueIdx;
20              int biggestMaxDistance = maxAbsValueIdx + 1 > maxDistanceToBack ?
21                                       maxAbsValueIdx + 1 : maxDistanceToBack;
22              int shiftedInputDataLength = biggestMaxDistance * 2;
23              double pointToAdd = 0;
24              int idxShift = shiftedInputDataLength / 2 - maxAbsValueIdx;
25              Queue<Double> shiftedInputData = new Queue<double>(shiftedInputDataLength);
26              if (maxAbsValueIdx <= _dataSamplesToClassifyQueue.Count / 2)
27              {
28                  for (int pointIdx = 0; pointIdx < shiftedInputDataLength; pointIdx++)
29                  {
30                      if (pointIdx > (idxShift) &&
31                         pointIdx < (shiftedInputDataLength / 2 + maxDistanceToBack))
32                      {//use original data
33                          pointToAdd = originalInputData[pointIdx - idxShift];
34                      }
35                      else
36                      {//fill with artificial data
37                          if (_dataSamplesToClassifyPreStack != null &&
38                                _dataSamplesToClassifyPreStack.Count > 0)
39                              pointToAdd = _dataSamplesToClassifyPreStack.Pop();//use some
                                    from calibration
40                          else
41                              pointToAdd = 0;
42                      }
43                      shiftedInputData.Enqueue(pointToAdd);
44                  }
45                  _dataSamplesToClassifyQueue = shiftedInputData;
46                  _dataSamplesToClassifyPreStack.Clear();//fresh for next sample
47              }
48              ...
```

Bei einer Sampling-Rate von 100 Hz zeichnet die Software in den 2.5 Sekunden des Blinzelvorganges 250 Datenpunkte auf. Zur Minderung dieser potentiell[153] übermäßigen Eingabedaten (*Feature-Reduktion*) kann der Benutzer unter ≪*DataPoints-for Classification*≫ wählen, wieviele Punkte als Eingabewerte für die Klassifikation verwendet werden sollen. Die endgültigen Input-Punkte ergeben sich aus dem Durchschnitt der dazwischen liegenden Werte.

Muster-ausdünnung

[153] hohe Sampling-Rate ⇝ viele Datenpunkte pro Zeit

Die Datensequenz des bereits angesprochenen, grau schattierten Beobachtungsbereiches der Klassifikation (siehe Abs. 5.3.3.5) muss also während einer Erkennung – über die Durchschnittsbildung – auch auf jene Menge von ≪*DataPoints for Classification*≫ reduziert werden. (siehe Quelltext 5.11)

Quelltext 5.11: `ClassifySample ()`: Klassifikation der beobachteten Daten

```
1   private Gesture ClassifySample()
2   {
3       ...
4       //prepare data: thinning for data-feature-reduction:
5       int pointsToTakeForClassification = chkTakeAllDataPoints.Checked ?
6                                 _recognDataSamplesCount :
                                      POINTS_TO_TAKE_IN_CLASSIFICATION_SAMPLE;
7       int pointsToAverageForOnePoint =
8         _recognDataSamplesCount / pointsToTakeForClassification;
9       int remainingPointsForAveraging = _recognDataSamplesCount %
            pointsToTakeForClassification;
10
11      double valueSum = 0;
12      Queue<double> tmpQueue = new Queue<double>(_dataSamplesToClassifyQueue);
13      Queue<double> dataToClassifyQueue = new Queue<double>(
            pointsToTakeForClassification);
14      int averagingPointsCount = 0;
15      while (pointsToTakeForClassification-- > 0)
16      {
17        for (int avgPointCounter = 0;
18          avgPointCounter < pointsToAverageForOnePoint; avgPointCounter++)
19        {//average as many points as calculated
20          valueSum += tmpQueue.Dequeue();
21        }
22
23        if (remainingPointsForAveraging-- > 0)
24        {//use up the remaining points one by one within the first final-points
25          valueSum += tmpQueue.Dequeue();
26          averagingPointsCount = pointsToAverageForOnePoint + 1;
27        }
28        else
29          averagingPointsCount = pointsToAverageForOnePoint;
30
31        dataToClassifyQueue.Enqueue(valueSum / averagingPointsCount);//average for final-
              point
32        valueSum = 0;
33      }
34
35      double[] inputArray = _dataSamplesToClassifyQueue.ToArray();//oldest data is at
            front
36      double[] inputDataArray = dataToClassifyQueue.ToArray();
37
38      //now classify with NNets:
39      Gesture freqGesture = ClassifyFrequencySample(ref inputArray);
40      Gesture dataGesture = ClassifyDataSample(ref inputDataArray);
41      return Gesture.NO_GESTURE;
```

```
42
43          }
```

Umsetzung der Diskontinuität

Direkt im Einsatz ist das Neuronale Netz nur zu bestimmten Zeiten. Das Training mit den Referenz-/Trainingsgestenmustern passiert z. B. nach dem Ende des Trainingsmodus' der Software. Die Zuordnung zu einer Geste in der Erkennungsphase startet erst nach Überschreitung des Schwellwertes.

Sie dauert dann maximal so lange bis die Länge der Trainingsbeispiele durchlaufen ist (gesteuert über `_possibleGestureHappendRecTimer`). In dieser Zeit schreitet der Beobachtungsbereich durch vom NIA gesendete Potentialdaten voran. Jeder neu hinzukommende Punkt löst eine Gestenklassifikation im Neuronalen Netz aus (über `ClassifySample()`). Die höchste generierte Wahrscheinlichkeit definiert die vom Benutzer vermutlich durchgeführte Geste. (siehe Quelltext 5.12)

Quelltext 5.12: Diskontinuierlicher Einsatz der Neuronalen Netze zur Klassifikation (in `RecognizeOnData(double time, double value)`)

```
1           if (_dataTrainingSetAlgLib != null &&//classify using neural net
2               !chkTrainMode.Checked)//not in trainingMode
3           {
4               CollectDataClassificationData(value);
5               if (_possibleGestureHappenedRecTimer > 0)
6                   _possibleGestureHappenedRecTimer--;
7
8               if (_possibleGestureHappenedRecTimer > 0 &&
9                _possibleGestureHappenedRecTimer < (double)1 / 3 * _recognDataSamplesCount)
10                  _possibleGestureHappenedRecTimer = -1;
11              else if (_possibleGestureHappenedRecTimer > 0 &&
12               _possibleGestureHappenedRecTimer < (double)2 / 3 * _recognDataSamplesCount)
13                  ClassifySample();//triggered a gesture..., let nnet decide which
14              else if (_possibleGestureHappenedRecTimer < 0)
15              {// stop recognition of possible gesture and show final results
16                  _possibleGestureHappenedRecTimer = 0;
17                  if (_recognizedGesturePossibility > _minimumGesturePossibility)
18                  {// show the best-fitting-gesture
19                      lblDataNetRec.Text = _recognizedGesture.ToString()
20                                           + " (" + _recognizedGesturePossibility.ToString("N3
                                                 ") + ")";
21                      //propagate the found gesture
22                      RecognizedNewBlink(this, _recognizedGesture);
23                  }
24                  else
25                  {
26                      Image bitmap = null;
27                      if (_recognizedGesture == Gesture.LEFT_EYE_BLINK)
28                          bitmap = Bitmap.FromFile(".\\leftBlinkNo.jpg");
29                      else if (_recognizedGesture == Gesture.RIGHT_EYE_BLINK)
```

```
30                          bitmap = Bitmap.FromFile(".\\rightBlinkNo.jpg");
31                      if (bitmap != null)
32                          picFace.Image = bitmap;
33                  }
34                  ...
35              }
36          }
37          else if (_collectTrainSamples)//user wants to capture a gesture.
                CollectDataClassificationData(value);
38          else//in trainingMode but currently not learning
39              _dataSamplesToClassifyPreStack.Push(value);
```

Sollte nach dem Ende der Klassifikation die erkannte Geste jener entsprechen, die zum „ausgelösten“ Schwellwert gehört (Qualifikation) und sollte die Wahrscheinlichkeit dieser erkannten Geste das einzustellende „Zuverlässigkeitsmaß“ neben ≪*min.-probability f. Rec.*≫ übersteigen, wird das Muster als *zuverlässig erkannt* klassifiziert. Die Vollzugsmeldung wird benutzerfreundlich ausgegeben. (siehe Quelltext 5.12 [Zeile 17ff])

Abbildung 5.15: Detail – „Erkennungsfeedback – Vollzugsmeldung“

Der Erkennungsprozess wird nämlich im Bereich unterhalb von ≪*TRAINING CAMP*≫ und oberhalb von ≪*TEST SUITE*≫ visuell (mit Bildern und Statusmeldungen) unterstützt (siehe GUI-Detail in Abbildung 5.15). Ebenso werden in den Gesture-Diagrammen die best-qualifizierten Gestenmuster der Klassifikation (Potentialmuster [links]) angezeigt (siehe GUI-Detail in Abbildung 5.16).

visuelles Feedback

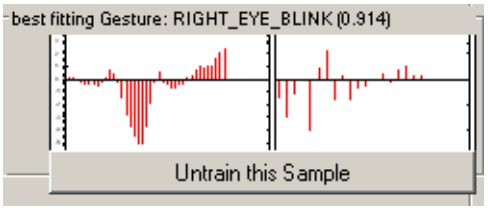

Abbildung 5.16: Detail – „Erkennungsfeedback – klassifiziertes Potential- (links) und Frequenzmuster (rechts)“

Als interessante Zusatzinformation existiert ein weiteres Gesture-Diagramm (ganz rechts unter dem Neuronalen Netz für Potentialdaten), zur Anzeige des Musters der global wahrscheinlichsten Geste. Ein Vergleich mit dem qualifizierten Ergebnismuster, kann bei unterschiedlicher Klassifikation (in der Programmentwicklung wertvolle) Einblicke in die Arbeitsweise des Netzes geben.

Im Programm ist die Klassifikationsleistung der Neuronalen Netze in ≪*NeuralNets Infos*≫ abzulesen (siehe GUI-Detail in Abbildung 5.17). Rechts ist dort das Neuronale Netz für die Potentialmustererkennung. Im oberen Bereich sieht man die Fehlerrate, welche das Netz bei Anwendung auf die Trainingsdaten aufweist. Darunter wird während eines Klassifikationsversuches[154] die wahrscheinlichste Geste inkl. Wahrscheinlichkeitsmaß („unmöglich“…0 bis 1…„sicher“) gezeigt.

Erkennungs-wahrscheinlichkeit

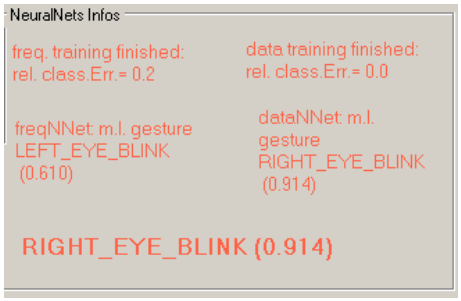

Abbildung 5.17: Detail – „Erkennungsfeedback – Die Ausgaben der neuronalen Netze“

Frequenzmusterklassifikation

Der NiaBlinkReader versucht, auch Frequenzinformationen für die Klassifikation zu nutzen. Das gesäuberte Signal wird dafür mittels *Fourier Transformation (FT)*[155] in die enthaltenen Frequenzen aufgespalten (*analysiert*). Die FT schafft – als Umsetzung des *Fourier Theorems* – nämlich die Darstellung einer beliebigen (auch nicht-periodischen) Funktion über Summation von sinusoidalen Funktionen. Kurz: *Jede Funktion kann als Summe von Sinusoiden beliebig genau angenähert werden.* Diese Sinusoide besitzen als sinusförmige Funktionen jeweils *Amplitude*, *Frequenz* und *Phase*. Die Amplitude gibt an, wie stark das analysierte Signal von der zugehörigen Frequenz „geprägt“ ist.

Fourieranalyse

(unendliche) Sinusoide als Basisfunktion

Die Verwendung der Fourier Transformation auf instationären Daten (wie im Fall der kontinuierlich gemessenen Potentialdaten) wird *Short Time Fourier Transform (STFT)* oder *Kurzzeit-Spektralanalyse* genannt. Die Dynamik birgt eigene Problemstellungen in sich. So ist es z. B. nicht sinnvoll eine FT auf großen, bewegten Datensätzen durchzuführen, weil sich ein reales Signal verändert und damit auch die enthaltenen Frequenzen pro Teilabschnitt unterschiedlich stark ausgeprägt sind; eine FT würde also nur „verschmierte“ Durchschnittsfrequenz-Komponenten liefern.

dynamische Daten

Die Lösung dieses Auflösungsproblems liegt darin, die FT auf kleineren Datenmengen durchzuführen. Die als *Fenster* bezeichneten Teilabschnitte sollen so groß gewählt

[154] ausgelöst nach Schwellwerterreichung
[155] siehe z. B. [Kluge 2006, S. 1ff]

werden, dass die zu findenden Frequenzen darin quasi-stationär angenommen werden können. Die Größe hängt verständlicherweise von den enthaltenen Frequenzen ab. Zur Erfassung hochfrequenter und daher möglicherweise stark lokalisierter Daten ist ein kurzes Fenster notwendig. Niederfrequente Anteile benötigen dagegen ein langes Fenster, um bei einer FT Berücksichtigung zu finden. Dieser Zwiespalt wird *Zeit-Frequenz-Unschärfe* genannt.[156]

Unschärfe-problem

Die Zeit-Frequenz-Unschärfe ist das größte Problem der STFT. Die Fenstergröße ist nämlich fixiert, da eine Vorhersage von Realdatenveränderungen zumeist unmöglich ist. Man legt durch Abschätzung interessierender Frequenzen von vornherein fest, worauf der Fokus der angewendeten Transformation gerichtet ist. Wenn die dominierenden Signalfrequenzen stark schwanken, wird die Fenstergröße also nur zeitweise passend sein; -ein Resultat des *stationären Charakters der Fourier Transformation*. Eine moderne Alternative für instationäre Frequenzanalyse bildet die sog. *Wavelet Transformation (WT)*.

fixe Fenstergröße

transiente Frequenzen

Der große Vorteil der WT liegt in der Flexibilität der Methode. Hochfrequente Details können ebenso extrahiert werden wie „lange“ Signalanteile (gleichzeitig!). Möglich ist dies durch den Einsatz unterschiedlich skalierter Basisfunktionen, den sog. *Wavelets*. Wavelets sind im Gegensatz zu den sinusoidalen Basisfunktionen der FT zeitlich lokalisiert. Durch Vergrößerung kann dieselbe Waveletform niedrige Frequenzen erfassen, aber an der zeitlich gleichen Signalposition – über verkleinerte Wavelets – ebenso die Details behandeln. Diese örtlich angepasste Signalauflösungsfähigkeit fehlt der FT wegen der fixen Fensterlänge komplett.

Wavelet Transformation

lokalisierte Basisfunktion

adaptive Auflösung

Informationen zur WT können z. B. in [Graps 1995] oder unter `http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html`[157] eingeholt werden. Die WT ist im Bereich der EEG-Klassifikation auf Frequenzbasis wegen der guten Feature-Extraktion und Schnelligkeit gebräuchlich[158].

Im NiaBlinkReader-Programm findet die Frequenzanalyse über die Fourier Transformation mit dem *Fast Fourier Transformation (FFT)*-Algorithmus aus der FFTW-Bibliothek[159] Verwendung. Um den C-Code dieser FFT-Implementierung in C# einsetzen zu können, wurden *Wrapper*-Funktionen[160], beschrieben unter `www.sdss.jhu.`

[156] siehe [Kluge 2006, S. 35ff]
[157] Stand Sept. 2009
[158] siehe z. B. [Pittner und Kamarthi 1999], [Jahankhani et al. 2006], [Subasi 2006] oder [Benimeli und Sharman 2007]
[159] `http://www.fftw.org` (Stand Sept. 2009)
[160] siehe Datei: `http://www.sdss.jhu.edu/~tamas/bytes/progs/fftwlib.zip` (Stand Sept. 2009)

`edu/~tamas/bytes/fftwcsharp.html`[157], eingesetzt (siehe Klasse `STFFT`).[161] Für die prinzipiell überlegene Wavelet Transformation konnten keine freien C#-Bibliotheken gefunden werden und eine eigene Implementation war im Kontext dieser Arbeit zu umfangreich.[162]

Fast Fourier-Transformation

Die FFT ist eine Computermethode und basiert daher auf einer diskreten Anzahl von *Abtastwerten* des Signals im Fenster. Je höher die Abtastrate (Sampling-Rate), desto mehr Details des Signals können in die Analyse einfließen. Aufgrund der Theorie zur FT *können nur Frequenzen im Signal erkannt werden, welche maximal der halben Abtastfrequenz entsprechen.* Diese Bedingung wurde von SHANNON formuliert und ist als *Abtasttheorem* bekannt. Die höchste im Signal enthaltene Frequenz wird *Nyquistfrequenz* genannt.[163]

Shannon-Theorem

Nyquist-frequenz

Zur Verdeutlichung des Abtasttheorems stelle man sich eine Abtastfrequenz von 100 Hz vor. Das sind 100 Punkte innerhalb jenes Intervalls, welches das Signal in einer Sekunde „durchläuft“. Die harmonische Schwingung mit der höchsten Frequenz, welche mit diesen Punkten richtig „aufgespannt“ werden kann, hat 50 Wellentäler und 50 -berge; -also 50 volle Schwingungen oder *Perioden* in der betrachteten Sekunde. Diese 50 Hz (*Nyquistfrequenz*) entsprechen, wie von SHANNON angegeben, der Hälfte der gewählten Abtastfrequenz.

Aliasfrequenzen

Wird die Abtastrate zu klein gewählt (*Undersampling*) kann der sog. *Alias*-Effekt in der FT zu nicht vorhandenen Fehlfrequenzen führen. Zum Ausgleich wird die Abtastfrequenz oft bewusst höher gewählt, als nach dem Shannon-Theorem notwendig; dieses *Oversampling* steigert allerdings den erforderlichen Rechenaufwand.[164]

Referenz-frequenzen

Die FT erlegt dem Signal ein Frequenzraster auf, das von der Höhe der Sampling-Rate und der Fenstergröße abhängig ist. Der Quotient aus Sampling-Rate und Anzahl der Punkte im verwendeten Fenster gibt den Rasterabstand zwischen den *Referenzfrequenzen* der FT. Nur für Signale mit Frequenzen, welche genau den Referenzfrequenzen entsprechen, wird eine FT im Stande sein, diese isoliert zu zeigen.[165] (siehe Abbildung 5.18)

Leakage-Effekt

Aus der Rasterung resultiert ein weiteres Hindernis bei Anwendung der (ST)FT: *phasenverschobene Frequenzanteile.* Die Verschiebung ist dabei auf die Referenzfrequenzen bezogen. Durch eine nicht exakte Abstimmung des Abtastvorganges auf die

[161] für eigene Implementationen der FT siehe z. B. [Arndt 2009, S. 409ff]
[162] für eigene Implementationen der WT siehe z. B. [Arndt 2009, S. 549ff]
[163] siehe [Kluge 2006, S. 29ff]
[164] siehe [Kluge 2006, S. 29ff]
[165] siehe `http://www.dspdimension.com/admin/dft-a-pied` (Stand Sept. 2009)

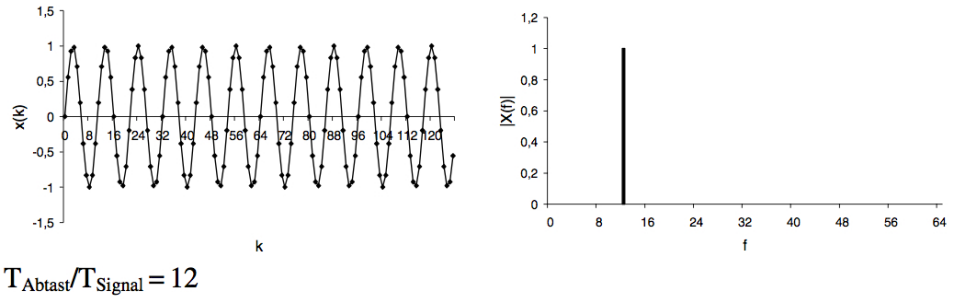

Abbildung 5.18: Exakte Frequenzbestimmung ohne *Leakage*-Effekt im Spektrum beim phasengleichen Signal [Kluge 2006, S. 31]

vorherrschende Periodizität des Signals, zeigt die Analyse den sog. *Leakage*-Effekt. Da die realen Frequenzen von der FT nicht genau erfasst werden, teilt sich dabei die Intensität dabei auf benachbarte Referenzfrequenzen auf (siehe Abbildung 5.19 zum Vergleich mit dem nicht verschobenen Signal in Abbildung 5.18).

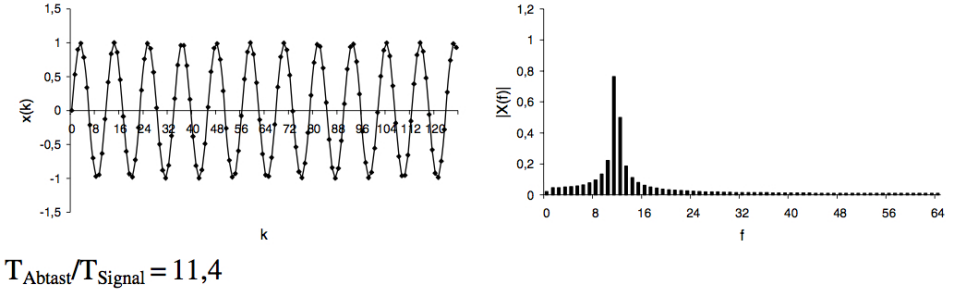

Abbildung 5.19: Ausgeprägter *Leakage*-Effekt im Spektrum durch phasenverschobenes Signal [Kluge 2006, S. 31]

spektrale Aufspreizung

Diese *spektrale Aufspreizung* entsteht dadurch, dass an den Intervallgrenzen das Signal seine Periode noch nicht beendet hat, die Frequenzanalyse der FT jedoch auf dieser Annahme beruht (*Stationarität der FT*); d. h. nach einer Rücktransformation (über die *Inverse Fourier Transformation*) würde das Signal am Ende die Periode fälschlicherweise vollenden. Dies käme dann einem *Signalsprung* (zum Ursprungssignal) an dieser Stelle gleich.[166]

Fensterfunktionen

Der Leakage-Effekt tritt bei realen Daten immer auf, da eine ständige Phasengleichheit aller im Signal beinhalteten Frequenzen höchst unwahrscheinlich ist. Das Problem kann durch Anwendung von „glockenförmigen“ *Fensterfunktionen* minimiert werden. Diese Kurven (siehe Abbildung 5.20, links) dämpfen, im Gegensatz zu Rechteckausschnitten, das Eingangssignal an den Rändern des Fensters gegen

[166] siehe [Kluge 2006, S. 29ff]

Null (siehe z. B. Abbildung 5.21, links). Da die Signalsprünge durch eine Frequenzaufspreizung am Rand auftreten, kann die Wahl einer passenden Fensterfunktionen eine Verminderung der Abweichungen bewirken. Vergleicht man das Spektrum in Abbildung 5.19 (rechts), sowie jenes desselben, jedoch mit einem *von-Hann*-Fenster multiplizierten Signals in Abbildung 5.21 (rechts), so ist eine merkliche Verbesserung auszumachen.

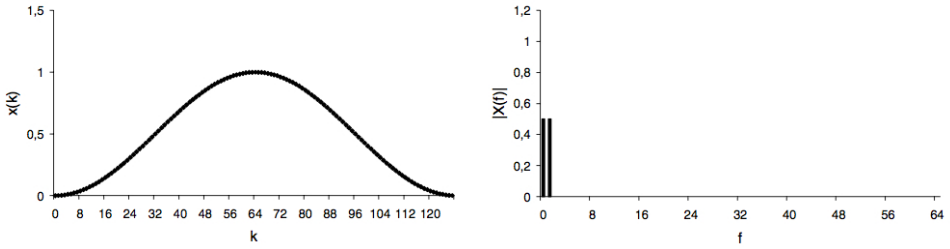

Abbildung 5.20: Die *von-Hann*-Fensterfunktion: $w(k) = 0.5 - 0.5 \cdot \cos(\frac{2\pi k}{M})$ [Kluge 2006, S. 32]

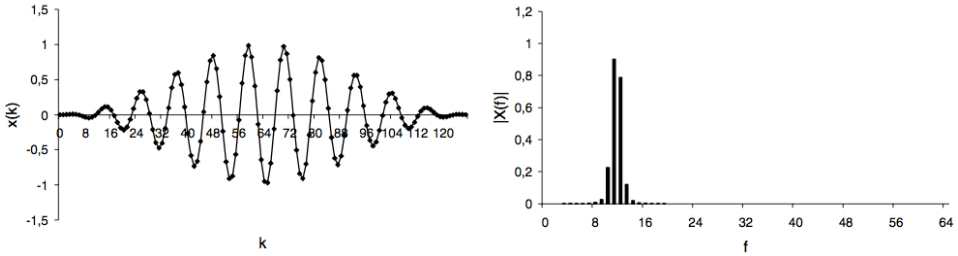

Abbildung 5.21: Verringerung des *Leakage*-Effekts im Spektrum nach Anwendung einer *von-Hann*-Fensterfunktion [Kluge 2006, S. 33]

Fensterüberlappung

Um eventuelle, nur an den Rändern der betrachteten Fenster auftretende Schwingungsanteile nicht durch die Anwendung einer Fensterfunktion auszulöschen, wird üblicherweise eine Überlappung der im Datenstrom aufeinander folgenden STFT-Intervalle eingeführt. Eine 50%-Überlappung bedeutet in diesem Fall, dass in der vorangegegangenen FT eventuell gedämpfte Randbereiche, in der nächsten – weil nun in der Mitte liegend – vollen Eingang in die Frequenzanalyse finden.[167]

Überlappung⇝ Oversampling

Auf der Webseite *The DSP Dimension*[168] wird von üblichen Überlappungen der FT-Fenster von (mind.) 75 % gesprochen. Die Überlappung entspricht einem Oversampling und je näher die Fenster aneinander liegen, desto genauer kann die wahre Frequenz im Signal extrahiert werden. Allerdings steigt mit steigender Überlappung

[167] siehe [Kluge 2006, S. 38]
[168] siehe `http://www.dspdimension.com/admin/pitch-shifting-using-the-ft` (Stand Sept. 2009)

auch die Anzahl der durchzuführenden Transformationen und damit der Rechenaufwand.

Das Programm begegnet den obigen Überlegungen mit einem – auf das Problem angepassten – Codes der Seite `http://sites.google.com/site/mikescoderama/pitch-shifting`[169]. Der adaptierte Quelltext zur Frequenzanalyse befindet sich in den Klassen `FrequencySeperation` sowie `FrequencyFilter`. Zum Verständnis der Grundlagen dieses Quelltextes sollten die Erklärungen auf *The DSP Dimension*[170] studiert werden. Als Fensterfunktion wird ein *von-Hann*-Fenster verwendet.

Frequenzen:
α, β, γ, δ, θ

Die Versuche mit der Frequenzanlyse gingen im Programm dahin, dass das Signal in den benannten EEG-Frequenzen (siehe Abs. 2.1, S. 10) dargestellt werden kann. Dafür muss neben ≪*do Frequency Seperation*≫ die Auswahl getroffen werden. Die Änderungen der einzelnen Frequenzen über die Zeit werden dann im unteren Teil als dynamisches Balken- (siehe GUI-Detail in Abbildung 5.22) sowie als Frequenz-Liniendiagramm angezeigt. Die, der Fourieranalyse zugrundeliegende, Fenstergröße entspricht ungefähr der Länge eines Blinzelvorganges (dauert 2-3 Sekunden; festgelegt über *SampleCollectionDuration* aus dem Trainingsmodus). (siehe Quelltext 5.13)

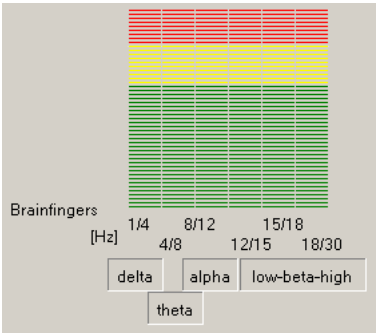

Abbildung 5.22: Detail – extrahierte „Brainfingers“

Quelltext 5.13: Behandlung der Brainfingerszerlegung
(in `FrequencyAnalysis(long timerPosition, long rawData)`

```
1  //samplerate * duration of blink = samples to analyze:
2  int currSampleSize = _alignedSamplingRateHz *
3                       (int)((float)trackBarSampleCollectionDuration.Value / 10);
4  if (_doFrequencyAnalysis && !chkTrainMode.Checked || _inCalibrationMode)
5  {
6     if (_fourierDataQueue.Count >= currSampleSize &&
7         _fourierShiftCounter > 10)
8     {
```

[169] Stand Sept. 2009
[170] siehe `http://www.dspdimension.com/admin/dft-a-pied` (Stand Sept. 2009) sowie `http://www.dspdimension.com/admin/pitch-shifting-using-the-ft` (Stand Sept. 2009)

```
9        _fourierShiftCounter = 0;
10       while(_fourierDataQueue.Count> currSampleSize)
11          _fourierDataQueue.Dequeue();
12       FrequencyProcessing.FrequencySeperation.GetFrequencies(currSampleSize,
13                                             currSampleSize, 1,
14                                             (float)_alignedSamplingRateHz,
15                                             _fourierDataQueue.ToArray(), true);
16    }
17    else
18       _fourierShiftCounter++;
19 }
```

Eine weitere Anwendungsidee für das Frequenz-Spektrum im Signal beinhaltete die Filterung des Signals. Die Eigenschaft der Fourier Transformation auch invertierbar zu sein, legt die Möglichkeit nahe, das Signal von den nicht interessierenden Frequenzen zu bereinigen. Über die *Inverse Fourier Transformation* gelangt man damit zu einem „reineren“ Datensignal. (siehe Quelltext 5.13)

Idee: Frequenzfilter

Quelltext 5.14: Durchführung der Filterung über Frequenzen (in `FrequencyAnalysis(long timerPosition, long rawData)`)

```
1  //samplerate * duration of blink = samples to analyze:
2  int currSampleSize = _alignedSamplingRateHz *
3                       (int)((float)trackBarSampleCollectionDuration.Value / 10);
4  ...
5  //frequency filter with STFT
6  if (_useFrequencyFilter &&
7      _fourierQueue.Count >= currSampleSize)
8  {
9     while (_fourierQueue.Count > currSampleSize)
10       _fourierQueue.Dequeue();
11    _freqFilteredData = _fourierQueue.ToArray();
12    FrequencyProcessing.FrequencyFilter.Filter(currSampleSize, currSampleSize,
13                                             _FFToverlapShare, (float)_alignedSamplingRateHz,
14                                             _freqFilteredData, false);
15    if (_useFrequencyFilter &&
16        _fourierProcessedRawDataList != null)
17        _filteredReadCounter = FOURIER_SAMPLE_SIZE;
18        _fourierQueue.Clear();
19 }
```

Störende Frequenzen sind in jedem Fall um 50 Hz, da hier die Frequenz des europäischen Stromnetzes Artefakte bildet. Weil das betrachtete Blinzeln hauptsächlich auf EOG Quellen zurückgeht, wären in dieser Anwendung die Frequenzen im unteren Bereich bis 4 Hz von Interesse. Auch die Betrachtung des Blinzelmusters selbst, deutet auf diesen Frequenzbereich hin; der ausgeprägte Wellenberg lässt sogar eine dominierende Frequenz unter 1 Hz vermuten. (siehe Abs. 3.1)

Artefaktbehandlung

Nach Auswahl im Programm neben ≪*show Freq.-filtered rawData*≫ und zusätzlicher Wahl der einzubeziehenden Frequenzen in ≪*Used Freqs*≫, erscheint im Frequenz-Liniendiagramm das (frequenz)gefilterte Signal. Es wird eine FT-Fensterüberlappung von 75 % verwendet. Die Fensterlänge gleicht wieder der Länge des Blinzelvorganges. (siehe GUI-Detail in Abbildung 5.23)

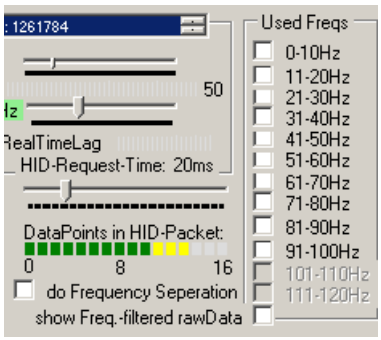

Abbildung 5.23: Detail – Frequenzfiltereinstellungen

Das gefilterte Signal könnte durchaus sehr gut für die Klassifikation der Blinzelmuster über Neuronale Netze verwendet werden, nur ergibt sich durch die Filtereigenschaften (Verwendung eines 2-3 Sekunden Musters; Rechenaufwand) eine zeitliche Verzögerung, welche die Klassifikation erst spät zulässt. Im Sinne eines wünschenswerten, direkten Feedbacks wird der *kontinuierliche* Frequenzfilter im Programm demnach nicht für Klassifikationsaufgaben eingesetzt. Die Implementierung der bereits angesprochenen *Wavelet Transformation* zur Frequenzanalyse könnte die Verzögerung reduzieren und die Idee des frequenzgefilterten Signals praktisch nutzbar machen.

Schwellwert-kopplung

Ein weiterer Ansatz, die Frequenzdaten für die Klassifikation auszunutzen, lag in der Kopplung mit dem bereits bekannten Schwellwert. Dadurch wird nur eine zeitweise Berechnung erforderlich und somit die Rechenlast gemildert. Das Prinzip folgt wiederum Abs. 5.3.3.5. Die überwachten Bereiche des Datensignals werden, nach der Überschreitung des Potentialdaten-Schwellwertes aus der Trainingsstatistik, mittels Fourier Transformation in das enthaltene Spektrum zerlegt. Die signifikanten Frequenzen werden von einem Neuronalen Netz auf mögliche Gesten getestet.

diskontinuierliche Analyse

Feature-Dekorrelierung

Mit *signifikanten Frequenzen* sind hier jene bezeichnet, welche die wesentliche Charakteristik des Blinzelvorganges beschreiben. Diese Frequenzen werden mittels einer *Principle Component Analysis* (PCA) nach der Trainingsphase zu ermitteln versucht. Die PCA ist eine statistische Methode zur *Dekorrelierung*, wobei die Variablen nach ihrem Beitrag zur Varianz im Signal geordnet werden. Die Varianz ist ein Maß des Informationsgehaltes. Ein konstantes Signal schwankt z. B. nicht (Varianz

Varianz ⇝ Information

= Null), weshalb auch das „Überraschungspotential“ des Signals Null ist. Anders ausgedrückt *bringt der fortlaufende Datenstrom eines konstanten Signals keine neuen Informationen.*

Feature-Reduktion

Es zeigt sich, dass zumeist nur wenige Variablen den Großteil der Gesamtvarianz verursachen. Bei Vernachlässigung der „unwichtigeren“ Komponenten geht daher wenig Information unter, der Verarbeitungsaufwand wird allerdings merklich reduziert. Ebenso verringern sich auch die nicht-signifikanten Störquellen im Signal. Diese Tatsache versucht man mit einer PCA auszunutzen. Mathematisch ausgedrückt beinhaltet PCA *„das lineare Transformieren der Daten in die Richtungen der Eigenvektoren durch eine Transformationsmatrix“*[171].

Implementiert wurde PCA (wie auch schon die Neuronalen Netze) über die Klassen[172] der ALGLIB[173]-Bibliothek.

Koordinaten-drehung

Die Transformationsmatrix wird im Zuge der PCA aus den Trainingsdaten ermittelt und definiert geometrisch eine Drehung. In der Erkennungsphase werden die Frequenzen (Vektoren) aus der Fourier Transformation mit dieser Matrix so transformiert, dass am Beginn des Resultatvektors die „wichtigsten“ Frequenzen zu liegen kommen.

Neuronales Netz

Im Programm werden daraus die ersten Zehn als Eingaben des, für Frequenzen zuständigen, Neuronalen Netzes (wieder ein *Ensemble*) verwendet. PCA ist somit das Mittel für eine Feature-Reduktion bei der Frequenzklassifikation. Der Einsatz von PCA kann im Programm durch Abwahl neben ≪*use PCA for freq.-recognition*≫ deaktiviert werden. Dann werden die Frequenzen der FT direkt für die Klassifikation verwendet.

experimentelle Software

Die Idee war, dass dieses Neuronale Netz in Kombination mit jenem der Potentialdaten-Mustererkennung eine robuste Erkennung des Blinzelvorganges ermöglicht. Leider funktioniert die Erkennung auf Frequenzbasis nicht zuverlässig, weshalb die Ausgaben dieses Netzes nur zu Informationszwecken unter ≪*NeuralNets Infos*≫ bzw. das erkannte Muster im Gesture-Diagramm (rechts) angezeigt werden. Auswirkungen auf die Gestenerkennung hat dies jedoch keine (mehr).

5.3.3.6 Testen der Zuverlässigkeit

Feedback im Betrieb

Befindet sich das Programm nicht im ≪*Training Mode*≫ und wurde bereits trainiert,

[171] siehe [Paetz 2006, S. 48]
[172] `http://www.alglib.net/dataanalysis/principalcomponentsanalysis.php` (Stand Sept. 2009)
[173] `http://www.alglib.net` (Stand Sept. 2009)

so kann die Gestenerkennung ausgelöst werden. Es ist nur notwendig das Potentialsignal über die Schwellwerte (grün schattiertes Band ober- und unterhalb der Nullline im Potentialdaten-Liniendiagramm) zu bewegen, um einen Klassifikationsversuch auszulösen. Die Ergebnisse können über die Bildschirmausgaben verfolgt werden.

stufenweises Training

Unbefriedigende Klassifikationsleistungen können eventuell durch „weitertrainieren" verbessert werden; dazu ist nur der abermalige Eintritt in den Trainingsmodus nötig. Nach dem Verlassen werden *alle* bisherigen Trainingsbeispiele neu trainiert.

False Positives

Da die Schwellwertübertretung ebenso über nicht-Blinzelvorgänge entstehen kann, muss die fehlertolerante Mustererkennung der Neuronalen Netze diese Falschauslösungen erkennen. Die entsprechende Einstellung des Zuverlässigkeitsmaßes neben *≪min. probability f. Rec.≫* soll dahingehend verändert werden, wie „passend" die Muster für eine gültige Gestenerkennung sein müssen.

False Negatives

adaptive Zuverlässigkeit

Zu beachten gilt, dass die Einstellung hier schwierig zu treffen ist, da die Ausgaben Neuronaler Klassifikations-Netze auf Wahrscheinlichkeiten beruhen. Weiters sind die jeweils vorherrschenden Trainingsituationen (Umgebung, Benutzer) unterschiedlich sind. Eine Zuverlässigkeitsanforderung, welche gewöhnlich gute Klassifikationsleistungen erbringt, wird unter widrigen Umständen fälschlich auch gewollt korrekte Gesten ausschließen. Die Justierung kann vom Benutzer des Programmes deshalb jederzeit (subjektiv) adäquat vorgenommen werden.

In *≪Test Suite≫* existiert ein rudimentäres, automatisches Testprogramm, welches die Klassifikationsgenauigkeit aufgrund der Ausgabe von Teststatistiken erkennen lässt. Nach der Betätigung von *≪Start Test≫* wird mit einer vollen Fortschrittsanzeige unterhalb des zu testenden Auges (*≪LEFT≫*/*≪RIGHT≫*) auf das erwartete Ereignis hingewiesen. Solange der Schwellwert nicht erreicht wurde, bleibt die Software im Zustand *≪Pending≫*. Klassifikationen können in *Success* oder *Failure* enden. Durch *≪Stop Test≫* wird der Test beendet und eine Gesamtstatistik angezeigt.

6 Benutzertests mit dem OCZ NIA

NIA ⇝ sensibel

Versuche mit verschiedenen Testpersonen haben einige Schwächen des Systems aufgezeigt. Die Ströme im menschlichen Körper sind äußerst klein und Messeinrichtungen müssen entsprechend sensibel sein. Diese Empfindlichkeit erhöht jedoch auch die Störanfälligkeit in Alltagssitutionen.

NIA ⇝ störanfällig

Der NIA wird schnell unkontrollierbar wenn äußere Einflüsse wie elektromagnetische Strahlung, Kabelbewegungen am Gerät, Verrutschen der Stirnelektroden, Änderungen des elektrischen Hautwiderstandes (z.B durch Schweiß) etc. auftreten. Dadurch liefert der NIA Potentialdaten an die Software, die nicht gewollt bzw. steuerrelevant sind.

NiaBlinkReader

Hauptsächlich wurde in den Benutzertests die NiaBlinkReader-Software getestet. Die Software kann zufällige, kurzzeitige, fehlerbedingte Ausschläge nicht erkennen. Sie entscheidet prinzipiell nur, wie ähnlich ein aufgetretenes Muster den Trainingsdaten ist. Einem durch Störquellen ausgelösten Rauschen kann durch die Signalverarbeitungsmöglichkeiten der Software großteils jedoch sehr wohl begegnet werden.

6.1 Versuchserkenntnisse

Blinzeln nicht immer „leicht“

Überraschenderweise ergaben die Versuche, dass separates Augenblinzeln keineswegs jeder Person „möglich“ ist. Der Grundgedanke der Softwareentwicklung - „natürliche, charakteristische Geste“ - scheint nicht allgemein gültig zu sein. Ob jene Personen durch längeres Üben der Geste Fortschritte machen würden, muss erst erprobt werden.

Benutzerabhängigkeit bestätigt

Auch wurde durch die Tests belegt, dass die Blinzelmuster der Personen sehr unterschiedlich aussahen. Das bestätigt den Einsatz der höchst individualisierbaren Neuronalen Netze für die Klassifikationsaufgabe. Schwierig ist allerdings, dass die Probanden teilweise gar keine charakteristischen Muster „erzeugen“ konnten (schwache/keine/willkürliche Ausschläge). Die Software besitzt dafür Signalverarbeitungseinstellungen, welche eine progressive Potentialverstärkung ermöglichen. Aufgrund

dieses erst kürzlich gezeigten Benutzer-Problems, muss die Funktionalität erst evaluiert werden. Generell muss akzeptiert werden, dass wenn selbst der Mensch kein Muster erkennt, der Computer wahrscheinlich ebenfalls keine Klassifizierung durchführen kann. Möglicherweise ist es auch hier eine Frage der Übung, wie „gut“ die jeweiligen Blinzelmuster ausgeprägt sind.

Neuronale Netze gut geeignet

Es zeigte sich, dass das Neuronale Netz der Potentialdaten die richtige Blinzelseite erkennt, die Zuverlässigkeitsschwelle oft jedoch nicht erreicht wird. Es sollten weitere Mechanismen erarbeitet werden, welche die Klassifikation bestätigen und damit die Zuverlässigkeit steigern. Der Ansatz über die bereits angesprochene Frequenzanalyse der Gestenmuster zielt darauf ab. Weiters könnten geometrische Algorithmen klassifizierungsfähige Features herausarbeiten.

Ermüdung

Die Versuchspersonen gaben teilweise an, dass das im Versuch häufig auszuführende Blinzeln für sie ermüdend wirkte. Einerseits kann dafür die Intensität des Versuchsaufgabe verantwortlich gemacht werden, andererseits möglicherweise die fehlende Übung. Jeder Muskel kann jedoch durch Training auf die Belastungen angepasst werden und in realen Anwendungen sollte die Blinzelhäufigkeit/-frequenz geringer sein.

Umstände für Qualität verantwortlich

Das Rauschverhalten des NIAs war ein offensichtliches Problem. Es scheint nicht nur die Umgebung für ein verrauschtes Signal verantwortlich zu sein, sondern auch der Benutzer selbst. Anscheinend laden sich Personen elektromagnetisch unterschiedlich stark auf, sodass der NIA in der gleichen Umgebung oft andere Signalqualitäten zeigt. Die Signalverarbeitungseinstellungen der Software sind deshalb individuell an Räumlichkeit und Person anzupassen.

Unkontrollierbarkeit

Einige der Probanden gaben zeitweise ein unkontrolliertes Signalverhalten an. Der NIA lieferte dabei Signale, die physiologisch nicht offensichtlich waren. Diese Störungen lösen teilweise ungeplante Klassifikationen aus; ein „Tastensperren“-Mechanismus (z. B. Druckknopf) könnte hier großteils Abhilfe schaffen. Der Benutzer würde die Geste erst durchführen/auslösen („entsperren“), wenn er die Kontrollierbarkeit erkennt. Damit klassifiziert die Software nur noch tatsächliche Steuer-Gesten.

Hardware $\rightsquigarrow$ limitierender Faktor

Die Störanfälligkeit der Hardware ist somit eines der größten Probleme, da ihr nicht leicht vom Benutzer oder über Softwaremethoden begegnet werden kann. Es bleibt idealerweise eine Aufgabe des Hardware-Herstellers, zukünftige Geräte robuster zu machen. Nachdem die Probleme nun vielfach erkannt wurden, sollte die Ursachenfindung bzw. deren technische Beseitigung nicht allzu schwierig sein.

6.2 Generalisierte Auffälligkeiten

Abgeleitet aus den durchgeführten Benutzertests und Recherchen zum NIA werden nachfolgend kurze Beschreibungen und Erklärungsversuche von zu erwartenden Resultaten gegeben. Einerseits basieren diese auf der Hardware, andererseits auf der Software. Sie können jedoch ebenso als prinzipielles Problem der BCI-Umsetzung im NIA gedeutet werden.

6.2.1 BCI-Methodik-induzierte Probleme

Übungsabhängigkeit
persönliche Fähigkeiten
Motivation

Generell kann davon ausgegangen werden, dass das Erlernen der Steuerung über Brainfingers (insbesondere die EEG-basierenden) einiger Übung bedarf. Manche Personen werden zwar durchaus schnelle Erfolge verbuchen können, andere erzielen hingegen nie eine befriedigende Leistung. Die Fortschritte hängen dabei sehr von der Lern-Motivation des Benutzers ab; *wer das Gerät nicht benutzt, wird es auch nicht beherrschen können.* Die Charakteristik der „Bedienung" von Brainfingers dürfte ähnlich jener von „normalen Gliedmaßen" sein; d. h. *wenn gewisse Tätigkeiten einmal erlernt sind, ist die Ausführung zumeist mühelos.*

6.2.2 Hardwareabhängige Probleme

Erdungsproblem

Problematisch war, dass während der Benutzung die Metall-Interfacebox des NIAs berührt werden muss, um überhaupt mit den Signalen arbeiten zu können. OCZ erklärt das notwendige Vorgehen bei solchen „Erdungs-" und Signalinterferenzproblemen unter `http://www.youtube.com/watch?v=4yOSPTpc63E`[174]. Sollte die Berührung des Metalls während der Benutzung nicht erwünscht bzw. möglich sein, wird geraten keine anderen Stromgeräte in der Nähe zu betreiben oder zu berühren. Diese scheinen für die Interferenzen verantwortlich zu sein.

BAT, der Hersteller des Cyberlink Brainfingers Systems, erwähnt in [Bra 2004, S. 10] und [Marler et al. 2006, S. 2 und 5] das Problem bei der Kalibrierung ebenfalls und bietet als eine der Lösungen ein Erdungsband an. Nach BAT tritt das Problem oft bei fehlender Erdung über das Stromnetz, beispielsweise bei Laptop-Computern, auf.

elektromagnetische Strahlung

Im offiziellen Forum[175] zum NIA wird über Lösungen diskutiert bzw. erklärt, wie das Problem entsteht. Ursächlich entsteht es durch die Tendenz des menschlichen Körpers, sich elektromagnetisch aufzuladen. Die *elektromagnetische Interferenz (EMI)* entsteht

[174] Stand Sept. 2009
[175] siehe `http://www.ocztechnologyforum.com/forum/showthread.php?t=48227` (Stand Sept. 2009) und

durch elektromagnetische Strahlung elektrischer Geräten in der Nähe des Körpers. Diese Aufladung (im mVolt-Bereich) leitet sich über das Kopfbandkabel in den NIA ab und die Signalaufbereitung (*Common Mode Rejection* siehe Abs. 4.3, S. 29) im Gerät wird gestört. Zudem soll das Kopfbandkabel eine Antennenwirkung haben und so die EMI zusätzlich verstärken. Im Handel erhältliche Antistatik-Armbänder schaffen keine Abhilfe, da sie mit dem rechtlich vorgeschriebenen 1 MOhm Widerstand die geringe Ladung nicht ableiten können. Im Forum wird vorgeschlagen, Kontakt zur Metallbox des NIAs über direkte Hautberührung zu erhalten bzw. soll ein *Telefon-* oder *Audiokabel* die Ableitung ermöglichen.

NIA-Nachfolger

Es wird vermutet, dass der kabellose Nachfolger des NIAs (siehe Abs. 4.3, S. 30) das Erdungsproblem lösen kann, weil dabei kein Kabel die Ladung in die Elektronik ableitet und somit keine Störung auftritt. Verfälscht die Aufladung allerdings schon die Aufzeichnung an den Elektroden, dürfte die Kabellosigkeit kein Ausweg sein.

Entfernung der EMI-Quelle

Gegen besonders starke EMI hilft jedoch auch die „Erdung“ nicht. Der NIA zeigt dann ein konstant überbordendes Rauschverhalten (*Signal/Noise*). Eine Gegenmaßnahme konnte im Betrieb mit dem NiaBlinkReader teilweise in der Erhöhung der Glättung durch den *gleitenden Durchschnitt* gesehen werden. Damit einhergehend muss jedoch auch die *Sample-Rate* erhöht werden, um die entstehende „Signalträgheit“ auszugleichen. Eine Entfernung der störenden elektrischen Quelle ist nach Möglichkeit aber das sicherste Mittel zur „Beruhigung“ der Ausschläge.

Kabel-bewegungen

Es fällt außerdem auf, dass bloße Bewegungen des Kopfband-Kabels nicht unbeträchtliche Signalausschläge zur Folge haben. Ein erfolgreicher mobiler Einsatz in „unruhigen“ Situationen erscheint dadurch schwierig. Dieses Problem sollte durch den kabellosen NIA wegfallen.

Kontakt-abhängigkeit

Reibung

Sperrfunktion

Schwerwiegend scheint die Empfindlichkeit des Elektroden-Haut-Kontaktes. Minimale Bewegungen der Elektroden auf der Haut lösen z. B. bereits, von tatsächlichen Potentialsignalen nicht zu unterscheidende, Ausschläge aus. In Hinsicht auf Situationen, wo ruckartige Kopfbewegungen vorherrschen, ist der Einsatz fraglich. Ein manuell zu steuernder „Freeze Mode“ könnte hier jedoch eine Lösung bieten – in unruhigen Situation wird das BCI-System z. B. durch Knopfdruck deaktiviert. BAT ist sich des Problems scheinbar auch bewusst und bietet für deren Cyberlink, wie erwähnt[176], Gel-Sensoren an. Sie sollen dieser „Verrutschgefahr“ entgegenwirken. Es kann andererseits davon ausgegangen werden, dass eben diese Reibung auch einen

Blinzel-Komponente

`http://www.ocztechnologyforum.com/forum/showthread.php?t=40139` (Stand Sept. 2009)

[176] siehe Abs. 4.1.1, S. 20

gewissen Beitrag zu den Blinzel-Signalen des NiaBlinkReaders leistet; wird doch beim Blinzeln allenfalls die Stirnhaut nach unten gezogen.

Leitfähigkeit

Weiters spielt die elektrische Leitfähigkeit der Haut eine Rolle. Generell kann das Signal über eine Feuchtigkeits-Hautcreme positiv beeinflusst werden. Die zeitlich bedingte Einwirkung derselben verändert allerdings wiederum das Hautmilieu. Selten kommt es zu nicht nachzuvollziehenden Ausschlägen, welche zumeist nach einigen Sekunden abklingen. Die bereits erwähnte[177] Passage im NIA-Handbuch betreffend der Einstellungsphase des *„Elektrolyt-Gleichgewichts zwischen den Mikroporen der Sensoren und der Haut“*[178] dürfte auf diese Milieuveränderungen gleichermaßen zutreffen.

Schweiß

Das Vorhandensein von Schweiß impliziert ähnliche Auswirkungen wie eine Feuchtigkeitscreme. Der potentiell hohe Elektrolytgehalt von Schweiß bewirkt eine besonders gute Leitfähigkeit. Wann bzw. ob überhaupt eine Auftrocknung stattfindet, ist ungewiss; in jedem Fall ändert sich die Signalübertragung.

6.2.3 Probleme mit der NIA Original-Software

Prozessorauslastung

Auffallend war eine hohe Prozessorauslastung bei Verwendung der Kontrollsoftware. Dies bestätigt, dass der NIA nur die Potentialdaten an den Computer sendet (siehe Schlussfolgerungen aus der Hardwareanalyse in Abs. 4.3, S. 30). Die Frequenzanalyse wird erst von der Computersoftware durchgeführt.

Mehrkernprozessoren

Schuette erklärt im bereits erwähnten[179] Interview[180], dass die Software *multithreaded* arbeitet. Somit kann die NIA-Signalverarbeitung auf heute üblichen Multikern-Prozessoren zumeist auf ungenutzte Kerne ausgelagert werden. Die Multithreading-Fähigkeit ist auch einer der wesentlichen Unterschiede zu der (älteren) Brainfingers-Software von BAT. Dies ermöglicht überhaupt erst den Einsatz für moderne Computerspielen.

mangelndes Feedback

Die NIA Software bietet dem Benutzer leider nicht die Unterstützung, welche möglich[181] und nötig wäre. Der Lernerfolg hängt einzig von den Anstrengungen des Benutzers ab. Dazu zählt u. a. die Experimentierfreudigkeit, Wege zu finden, welche den NIA nutzbar machen. OCZ gibt wenig Anleitung für schnelle Erfolge.

[177] siehe Abs. 5.1, S. 32
[178] siehe [OCZ 2008, S. 5]
[179] siehe Abs. 4.3, S. 25
[180] siehe `http://www.lostcircuits.com/mambo//index.php?option=com_content&task=view&id=32&Itemid=47&limit=1&limitstart=1` (Stand Sept. 2009)
[181] siehe Softwaresammlung des Cyberlink-Systems; Abs. 5.2, S. 37

7 Schlussbetrachtungen

7.1 Zusammenfassung

Forschungsarbeit

Diese Masterarbeit dient der Forschungstätigkeit am *Institut für Bauinformatik*[182] der *Technischen Universität Graz*[183]. Sie gibt generell Aufschluss über die Möglichkeit zur freihändigen Menüsystemsteuerung mit Brain-Computer-Interfaces (BCI). Von besonderem Interesse war dabei die Evaluierung der allgemeinen Möglichkeiten von kostengünstigen BCI-Geräten.

BCI-Menüsteuerung

EEG-Basis⇝ gut und teuer

Es wurde gezeigt, dass ein Einsatz von BCI für diverse Tätigkeiten durchaus sinnvoll ist. Speziell die Nutzung jener Systeme auf der Basis von Hirnsignalmessung über *Elektroenzephalografie (EEG)* verspricht, dank der langjährigen Erfahrungen in der Medizintechnik, hohes Potential. Leider sind diese EEG-Geräte aber dementsprechend teuer und entziehen sich daher (noch) dem Massenmarkt. Sollten die – zumeist EEG einsetzenden – BCI-Forschungseinrichtungen jedoch irgendwann marktreife Systeme anbieten können, werden, wie bei jeder nachgefragten Technologie, die Preise durch auftretende Konkurrenz fallen.

BCI – Useability

In dieser Arbeit wurde gezeigt, dass erfolgreiche BCI eine Kombination aus Hardware-, Software- und Benutzer-Fähigkeiten ist. Es wurde deutlich, dass nicht jede Person dazu in der Lage ist, BCI (schnell) zu erlernen. Besonders im Falle von EEG-BCI handelt es sich um die bewusste Steuerung interner physiologischer Vorgänge, deren Aktivierung einer ungewöhnlichen Aufgabe entspricht. Sie ist nämlich sehr individuell und kann von Dritten nicht leicht beobachtet/kontrolliert/unterstützt werden (Feedback). Der Benutzer muss selbst erkennen, wie das System *Gehirn-Computer* zusammenarbeitet.

Low Cost-BCI

Spielefokus

Biosignale

Für den verbreiteten Einsatz eignen sich derzeit erhältliche, *kostengünstige BCI-Geräte* der *Computerspiele-Branche*. Der Markt ist dort noch spärlich besetzt und die Systeme funktionieren über die Registrierung sowie Interpretation von *elektrischen Potentialen* im Bereich des Kopfes. Interessant erscheinen all diese Geräte gleicher-

[182] `http://bauinformatik.tugraz.at` (Stand Sept. 2009)
[183] `http://www.tugraz.at` (Stand Sept. 2009)

maßen, weil sie doch einiges versprechen; inwieweit sich die *Marketing-Aussagen* aber in der Praxis bewahrheiten, wird sich weisen.

OCZ NIA

Eine Sonderstellung bezieht der, in dieser Arbeit eingehend behandelte, *Neural-Impulse Actuator (NIA)* von *OCZ Technology Inc.* (OCZ) – ein Computereingabegerät zur unterstützenden Steuerung von Spielen. Wegen der „Abstammung" vom *Cyberlink-Brainfingers-System* des Unternehmens *Brain Actuated Technologies Inc.* hat die zugrundeliegende Technologie bereits eine gewisse Tauglichkeit nachweisen können. Der Cyberlink mit vormals militärischem Entwicklungshintergrund wird seit einigen Jahren erfolgreich im Bereich der *Computersteuerung* verwendet. Die Kompaktheit des Systems lässt einen *mobilen Einsatz* ebenso möglich erscheinen.

bewährte Technologie

Mix: EOG, EMG, EEG

Der Cyberlink gehört zu den höherpreisigen BCI-Systemen und nimmt an *drei Stirnelektroden* Biopotentiale auf – jedoch nicht nur klassische EEG-Daten. Vielmehr sind die Daten eine Mischung aus *Hirnstrom (EEG)*- und *Muskel (EMG)*- wie auch *Augen (EOG)-induzierten* Signalen. Die Erfolge des Cyberlinks am Gebiet der therapeutischen Medizin lassen hoffen, dass der NIA ähnlich „gut" funktioniert; -sind die beiden Systeme, wie in der Arbeit gezeigt, doch äußerst ähnlich.

OCZ liefert Spielesoftware

Die Software scheint den essentiellen Unterschied darzustellen. OCZ liefert im Vergleich nur eine sehr auf Spiele fokussierte Kontrollsoftware. Dementsprechend ist diese für sonstige Computersteuerung ungeeignet. Außerdem erschwert OCZ durch die zugrundegelegte Lernstrategie[184] dem Nicht-Spieler die Erlangung der Kontrolle.

Software-entwicklung

Nach Manifestierung dieser Software-seitigen „Mängel" wurde als Praxispart dieser Arbeit an einer „Ersatzsoftware" für den NIA zu entwickeln begonnen. Dafür mussten zu Beginn die abhängigen BCI-Wissensgebiete aufgearbeitet werden. Mit den theoretischen Grundlagen wurde versucht, die Signale des NIAs auszulesen und zu interpretieren. Wieder mit Blick auf die Idee der BCI-Menüsteuerung werden damit einfache Gesten (z. B. „Augenblinzeln") klassifiziert. Die Steuerung einfacher Menüs kann durch Unterscheidung weniger Befehle durchaus befriedigend erfolgen (man denke z. B. an Mobiltelefone). Zudem war die Forderung nach leichter/schneller Erlernbarkeit ein zentraler Aspekt der Softwareentwicklung.

Gesten-interpretation

Der Ansatz der nicht-manuellen Computersteuerung über Körper-Gesten, erscheint im Gegensatz zu reinen EEG-Systemen einfacher zu sein. Eine durchschnittliche Person kann gewisse Bewegungen aufgrund natürlicher menschlicher Eigenschaften bzw. durch Nachahmung erlernen. Das in dieser Arbeit beschriebene „Augenblinzeln"

[184] Der Benutzer muss sich der nicht offensichtlichen Funktionsweise der Software anpassen sowie das Lernen über Computerspiele.

sollte einer solchen Bewegung entsprechen. Zudem zeigt das low-cost BCI-System NIA von OCZ auf einseitiges Blinzeln eine charakteristische Reaktion.

Proof-of-Concept

Der gewählte Ansatz darf als initialer Entwicklungsvorstoß in der Erprobung des NIAs angesehen werden. Die Erweiterung und mögliche Umsetzung des Schemas in damit auszustattenden Menüsystemen kann, durch die Beschreibungen in dieser Arbeit sowie Studium des Programmquelltextes, leicht (auch auf verschiedenen Systemen) erfolgen.

System lernt vom Benutzer

schnell einsatzfähig

Einer der wichtigsten Ansatzpunkte in der Softwareentwicklung war, dass sich die Klassifikation wesentlich am Benutzer orientiert. So muss sich der Anwender nicht zur Gänze an das System anpassen, um es steuern zu können. Vielmehr werden individuelle Eigenheiten der Gestenumsetzung vom Klassifikationsalgorithmus toleriert bzw. können sie als herausstechende Merkmale die Erkennung sogar begünstigen. Der Sinn des Vorgehens lag in der Vermeidung einer zu langen Erlernphase (siehe Original-Software), da dies einen Demotivationsfaktor erzeugt. In einer vom Benutzer durchzuführenden, wenige Minuten dauernden Trainingsphase lernt die erstellte Software, wie dessen Gesten ausgeprägt sind. Diese „Vorführung“ der Gesten durch den Benutzer spiegelt die *Lernfähigkeit* des Systems wider; das eigene Lernerfordernis des Anwenders wird minimiert/delegiert.

Zuverlässigkeit

Fehlklassifikation

Sperrfunktion

Wie in Trainingssituationen üblich, bedeutet ein länger durchgeführtes Training in der Regel bessere Lernerfolge. Es liegt also am Benutzer, wie schnell das System einsatzbereit ist bzw. wie zuverlässig die spätere Gestenerkennung arbeitet. Beispielhaft sei angemerkt, dass die entwickelte Software bereits mit fünf Trainingsbeispielen pro Geste Erkennungswahrscheinlichkeiten von über 80% zeigte. Fälschliche Klassifikation von Nicht-Gesten soll durch den angewandten Algorithmus aus dem Bereich der *Computational Intelligence* verhindert werden. Eine gänzliche Fehlerfreiheit kann aber nie erreicht werden, denn bewusst ausgeführte Gesten gleichen eben öfters auch unbewusst ausgelösten Signalen. Eine mögliche – manuell einzuleitende – „Sperrung“ der Klassifikation bei Nicht-Benötigung der BCI-Funktion könnte Abhilfe schaffen.

Artefakte

Problematisch waren einige eher Hardware-seitig zu lösende Situationen. Der „Feinheit“[185] der verwendeten Biosignale ist es geschuldet, dass die Elektronik sehr sensibel sein muss. Äußere Umstände haben daher wesentliche Auswirkungen auf den NIA. Wegen Störquellen veränderte Signale werden allgemein *Artefakte* genannt. Artefaktbehandlung ist für jedes BCI-System ein entscheidendes Thema.

starkes Rauschen

Sehr unangenehm sind z. B. Störungen aus *elektromagnetischer Strahlung*. Stärkere Beeinflussungen durch entsprechende Elektrogeräte im Umfeld verändern die

[185] EEG-Potential-Amplituden liegen im Bereich $<100\,\mu$Volt

NIA-Signale so sehr, dass die resultierende „Unkontrollierbarket“ eine Gestenerkennung (ohne Gegenmaßnahme) aussichtslos macht. Als Ausweg wurde die Beseitigung der Quellennähe gefunden bzw. wird, wie in der Arbeit beschrieben[186], die Erhöhung der *Glättungsfunktion* vorgeschlagen. Wünschenswert wäre eine Lösung des *elektromagnetischen Problems* auf Hardware-Ebene.

„leichtere“ Probleme

Weiteren Problemen im Umgang kann leichter begegnet werden, da sie in der Sphäre des Anwenders liegen. Einige der, in dieser Arbeit erwähnten, sonstigen Artefakt-Ursachen können durch kreative Ideen beseitigt werden. Der Benutzer selbst läd sich z. B. schon bei relativ normalen elektromagnetischen Gegebenheiten genügend weit auf, um eine signifikante Interferenz mit den Biosignalen zu erzeugen. Solche geringen *Aufladungen* können in Leitungen mit geringem elektrischen Widerstand abfließen (etwa über Hautkontakt zum Metallgehäuse des NIAs). Die *Leitfähigkeit der Haut* kann über Hautcremes gesteuert werden, *signalstörende Bewegungen der Kabel* durch Kabellos-Technologie beseitigt werden. Andere Befestigungsarten fixieren die *Position* und minimieren die *Bewegungen der Elektroden*.

Aufladung

Feuchtigkeit

Bewegungen

Dies alles weist eine große Abhängigkeit des NIAs von äußeren Einflüssen aus. Deren Lösung obliegt eigentlich dem Hardware-Hersteller, deren Auswirkungen sind teilweise jedoch auch vom Benutzer zu regeln.

7.2 Ausblick

Potential

Die Vorteile und Möglichkeiten der Brain-Computer-Interface Thematik wurden aufgezeigt. Die Technologie beweist hohes Anwendungspotential für verschiedenste Interessenten. Problematisch sind derzeit die Stückkosten und die fehlende Marktreife der gut funktionierenden Systeme. Kostenmäßig interessantere Geräte sind funktionell (bewusst) eingeschränkt. Auch ihre frühe Marktzeit bedingt noch einige Verbesserungsmöglichkeiten der angebotenen Systeme; dies lässt in Anbetracht der Möglichkeiten doch auf Nachfolgemodelle hoffen.

Nachfolger

einfaches Konzept

Die Forderung nach einer gewissen Einfachheit für den BCI-Benutzer ist mit dem „Blinzelansatz“ in dieser Arbeit gelungen. Auch die passende Ausnutzung der NIA-Hardware-Fähigkeiten wurde damit erreicht.

mobiler Einsatz

Vor allem in Hinsicht auf einen mobilen Einsatz, der für reale Nutzungen enorme Bedeutung hat, muss die Hardware adaptiert werden. Der NIA wurde von OCZ mit

[186] hoher *gleitender Durchschnitt* und hohe *Sample-Rate*

anderen Motiven am Markt positioniert. Eine mobile Verwendung des Computerspiele-Zubehörs NIA ist für OCZ nicht relevant. Erst der Einsatz des NIAs wie in dieser Arbeit vorgeschlagen, lässt die Fehleranfälligkeit unangenehm zutage treten. Selbst stationäre Versuche zeigen teilweise eine zeitlich begrenzte Unkontrollierbarkeit, welche auf Benutzer bzw. Umgebung zurückzuführen sind. Eine, aus der angestrebten Mobilität folgende, ständige Umgebungsveränderung setzt den NIA darüber hinaus wechselnden Störquellen aus. Es müssen zukünftig Hardware-Vorkehrungen einfließen, sodass natürliche Störfaktoren der Mobilität in den Hintergrund treten.

Artefakt-behandlung

Ob der alleinige Einsatz des NIAs jemals eine hohe Zuverlässigkeit bieten kann, ist im Moment fraglich. Sicher ist, dass die Gestenerkennung prinzipiell funktioniert. Der NIA sollte in einer Kombination aus mobilen Eingabemöglichkeiten daher einen guten Beitrag leisten können, die zuverlässige Menüsteuerung zu ermöglichen. Hinsichtlich des ohnehin vorhandenen Kopfbandes, erscheint das Anbringen einer Kamera keine Schwierigkeit darzustellen. Mit ihrer Hilfe und visuellen Algorithmen, könnte das einseitige Augenblinzeln zusätzlich erkannt werden. Der, über Eye-Tracking-Methoden zu findende, Fokuspunkt des Benutzers könnte weitere nützliche Informationen zur Computersteuerung bieten.

Zukunft

Literaturverzeichnis

eia 1969

EIA Standard RS-232-C Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Data Interchange. August 1969

Arndt 2009

Arndt, Jörg: *Matters Computational – ideas, algorithms, source code.* `www.jjj.de/fxt/fxtbook.pdf`. Version: 07 2009

Bayliss 2001

Bayliss, Jessica D.: *A Flexible Brain-Computer Interface*, University of Rochester, New York, USA, Diss., 2001. `ftp://ftp.cs.rochester.edu/pub/papers/robotics/01.tr756.Flexible_brain-computer_interface.pdf`

Benimeli und Sharman 2007

Benimeli, Francesc ; Sharman, Ken: Electroencephalogram signal classification for brain computer interfaces using wavelets and support vector machines. In: *ESANN*, 2007, S. 361–366

Bra 2004

Brain Actuated Technologies Inc. (Hrsg.): *Brainfingers - User Manual.* 6.0. 1350 President Street, Yellow Springs, Ohio 45387 USA: Brain Actuated Technologies Inc., 2004. `http://www.brainfingers.com/V60Manual.pdf`

Darvas et al. 2004

Darvas, F. ; Pantazis, D. ; Kucukaltun-Yildirim, E. ; Leahy, R.M.: Mapping human brain function with MEG and EEG: methods and validation. In: *NeuroImage* 23 (2004), Nr. Supplement 1, S289 - S299. `http://dx.doi.org/10.1016/j.neuroimage.2004.07.014`. – DOI 10.1016/j.neuroimage.2004.07.014. – ISSN 1053–8119. – Mathematics in Brain Imaging

Ebner und Deuschl 2006

Ebner, Alois ; Deuschl, Günther: *EEG.* Thieme, Stuttgart, 2006. – ISBN 3–13–140101–X

Fatourechi et al. 2007

Fatourechi, Mehrdad ; Bashashati, Ali ; Ward, Rabab K. ; Birch, Gary E.: EMG and EOG artifacts in brain computer interface systems: A survey. In: *Clin Neurophysiol* 118 (2007), Mar, Nr. 3, S. 480–94. http://dx.doi.org/10.1016/j.clinph.2006.10.019. – DOI 10.1016/j.clinph.2006.10.019

Graps 1995

Graps, Amara: An Introduction to Wavelets. In: *Computing in Science and Engineering* 2 (1995), Nr. 2, S. 50–61. http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/99.388960. – DOI http://doi.ieeecomputersociety.org/10.1109/99.388960. – ISSN 1070–9924

He et al. 2006

He, Bin ; Hori, Junichi ; Babiloni, Fabio: Electroencephalography (EEG): Inverse Problems. In: *Wiley Encyclopedia of Biomedical Engineering* (2006), 04, S. 1–9

Jahankhani et al. 2006

Jahankhani, Pari ; Kodogiannis, Vassilis ; Revett, Kenneth: EEG Signal Classification Using Wavelet Feature Extraction and Neural Networks. In: *JVA '06: Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing*. Washington, DC, USA : IEEE Computer Society, 2006. – ISBN 0–7695–2643–8, S. 120–124

Junker et al. 1989

Junker, A.M. ; Schnurer, J.H. ; Ingle, D.F. ; Downey, C.W.: Brain actuated control of a roll axis tracking simulator. In: *Aerospace and Electronics Conference, 1989. NAECON 1989., Proceedings of the IEEE 1989 National*, 1989, S. 714–717 vol.2

Junker 1993

Junker, Andrew: *Brain-body actuated system*. U.S. Patent #5474082 issued Dec. 12, 1995. http://www.patentstorm.us/patents/5474082/description.html. Version: 1993 (filed)

Junker 1995

Junker, Andrew: *Brain-body actuated system*. U.S. Patent #5692517 issued Dec. 2, 1997. http://www.patentstorm.us/patents/5692517/description.html. Version: 1995 (filed)

Junker und Berg 2001

Junker, Andrew ; Berg, Christian R.: *Brain-Body actuated system*. U.S. Patent

#6636763 issued Oct. 21, 2003. http://www.patentstorm.us/patents/6636763/description.html. Version: 2001 (filed)

Junker et al. 2008a

Junker, Andrew ; Marler, Danise M. ; Hook, Bill: *Coherent Detected Periodic Brainwave Computer Control*. National Institute of Child Health and Human Development, R43 - Small Business Innovation Research Grants (SBIR) - Phase I, [1R43 HD 42942-01]. http://www.brainfingers.com/BATFinalNIH1.PDF. Version: 2008

Junker et al. 2008b

Junker, Andrew ; Sudkamp, Thomas ; Eachus, Todd ; Edmister, Evette ; Mikov, Terry ; Wegner, Jane ; Livick, Susan ; Heiman-Patterson, Terry ; Goren, Mark: *Hands-free Computer Access for the Severely Disabled*. National Institute of Child Health and Human Development, R43 - Small Business Innovation Research Grants (SBIR) - Phase I, [1R43 HD 39070-01]. http://www.brainfingers.com/BATFinalNIH2.PDF. Version: 2008

Kluge 2006

Kluge, Oliver: *Praktische Informationstechnik mit C#*. Springer-Verlag Berlin Heidelberg, 2006. – ISBN 978–3–54020812–9

Lippe 2007

Lippe, Wolfram-Manfred: *Soft-Computing: mit Neuronalen Netzen, Fuzzy-Logic und Evolutionären Algorithmen (eXamen.press)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2007. – ISBN 3540209727

Malmivuo und Plonsey 1995

Malmivuo, Jaakko ; Plonsey, Robert: *Bioelectromagnetism : Principles and Applications of Bioelectric and Biomagnetic Fields*. Oxford University Press, USA, 1995 http://www.bem.fi/book/in/bembook.zip/. – ISBN 0195058232

Marler et al. 2006

Marler, Danise M. ; Junker, Andrew ; Lustre, Breanna: *Training Tutorial*. 1350 President Street, Yellow Springs, Ohio 45387 USA: Brain Actuated Technologies Inc., 2006

Marler 2004

Marler, Danise M.: *Cyberlink: Computer Access for Persons identified with multiple Disabilities*, California State University, Northridge, Diplomarbeit, 08 2004. http://www.brainfingers.com/CyberlinkCaseStudy.pdf

Nelson et al. 1997

Nelson, W. T. ; Hettinger, Lawrence J. ; Cunningham, James A. ; Roe, Merry M. ; Haas, Michael W. ; Dennis, Leon B.: Navigating Through Virtual Flight Environments Using Brain-Body-Actuated Control. In: *Virtual Reality Annual International Symposium* 0 (1997), S. 30. http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/VRAIS.1997.583041. – DOI http://doi.ieeecomputersociety.org/10.1109/VRAIS.1997.583041. ISBN 0–8186–7843–7

Nickolls 2002

Nickolls, Peter: *Electroencephalogram (EEG)*. http://www.eelab.usyd.edu.au/ELEC3801/notes/electroencephalogram.pdf. Version: 2002. – Vorlesungsunterlagen für ELEC3801 Fundamentals of Biomedical Engineering

Niedermeyer und da Silva 2005

Niedermeyer, Ernst ; Silva, Fernando L.: *Electroencephalography - basic principles, clinical applications, and related fields*. 5. Lippincott Williams & Wilkins, Philadelphia, 2005. – ISBN 0–7817–5126–8

OCZ 2008

OCZ Technology Inc. (Hrsg.): *NIA neural impulse actuator - Installation Manual*. 860 E. Arques Ave., Sunnyvale, CA 94085 USA: OCZ Technology Inc., 2008. http://www.ocztechnology.com/manuals/German_NIA_web.pdf

Olejniczak 2006

Olejniczak, P.: Neurophysiologic basis of EEG. In: *Journal of clinical neurophysiology : official publication of the American Electroencephalographic Society* 23 (2006), June, Nr. 3, 186–189. http://dx.doi.org/10.1097/01.wnp.0000220079.61973.6c. – DOI 10.1097/01.wnp.0000220079.61973.6c. – ISSN 0736–0258

Paetz 2006

Paetz, Jürgen: *Soft Computing in der Bioinformatik: Eine grundlegende Einführung und Übersicht (eXamen.press)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2006. – ISBN 354029886X

Pittner und Kamarthi 1999

Pittner, Stefan ; Kamarthi, Sagar V.: Feature Extraction From Wavelet Coefficients for Pattern Recognition Tasks. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (1999), Nr. 1, S. 83–88. http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/34.745739. – DOI http://doi.ieeecomputersociety.org/10.1109/34.745739. – ISSN 0162–8828

Pregenzer und Pfurtscheller 1999

Pregenzer, M. ; Pfurtscheller, G.: *Frequency Component Selection for an EEG-Based Brain to Computer Interface*. IEEE Engineering in Medicine and Biology Society

Schad 2002

Schad, Lothar R.: Funktionelle Magnetresonanztomographie (fMRT) - Teil 1: Grundlagen und Messtechniken. In: *Der Radiologe* 42 (2002), 8, Nr. 8, S. 659–669. http://dx.doi.org/10.1007/s00117-002-0788-0. – DOI 10.1007/s00117–002–0788–0. – ISSN 0033–832X (Print) 1432–2102 (Online)

Scherer et al. 2007

Scherer, Reinhold ; Schloegl, Alois ; Lee, Felix ; Bischof, Horst ; Jansa, Janez ; Pfurtscheller, Gert: The self-paced graz brain-computer interface: methods and applications. In: *Computational Intelligence and Neuroscience* 2007 (2007), S. 9. http://dx.doi.org/10.1155/2007/79826. – DOI 10.1155/2007/79826

Subasi 2006

Subasi, Abdulhamit: Automatic detection of epileptic seizure using dynamic fuzzy neural networks. In: *Expert Systems with Applications* 31 (2006), Nr. 2, 320 - 328. http://dx.doi.org/10.1016/j.eswa.2005.09.027. – DOI 10.1016/j.eswa.2005.09.027. – ISSN 0957–4174

Thompson 2001

Thompson, Richard F.: *Das Gehirn: von der Nervenzelle zur Verhaltenssteuerung*. Spektrum Akademischer Verlag GmbH, Heidelberg, 2001. – ISBN 3–8274–1080–0

Übeyli 2009

Übeyli, Elif D.: Statistics over features: EEG signals analysis. In: *Computers in Biology and Medicine* 39 (2009), Nr. 8, 733 - 741. http://dx.doi.org/10.1016/j.compbiomed.2009.06.001. – DOI 10.1016/j.compbiomed.2009.06.001. – ISSN 0010–4825

Weicker 2007

Weicker, Karsten: *Evolutionäre Algorithmen.*. Bd. 2. 1. B.G. Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 2007. – ISBN 3519003627

Eidesstattliche Erklärung

Ich, Martin Krammer, Matrikel-Nr. 0330849, versichere hiermit, dass ich meine Masterarbeit mit dem Thema

Evaluierung der Möglichkeiten von kostengünstigen Brain-Computer-Interfaces

bezüglich der Eignung
zur freihändigen Bedienung von Computern

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Graz, den 15.November 2009

Martin Krammer

A Anhang

A Anhang

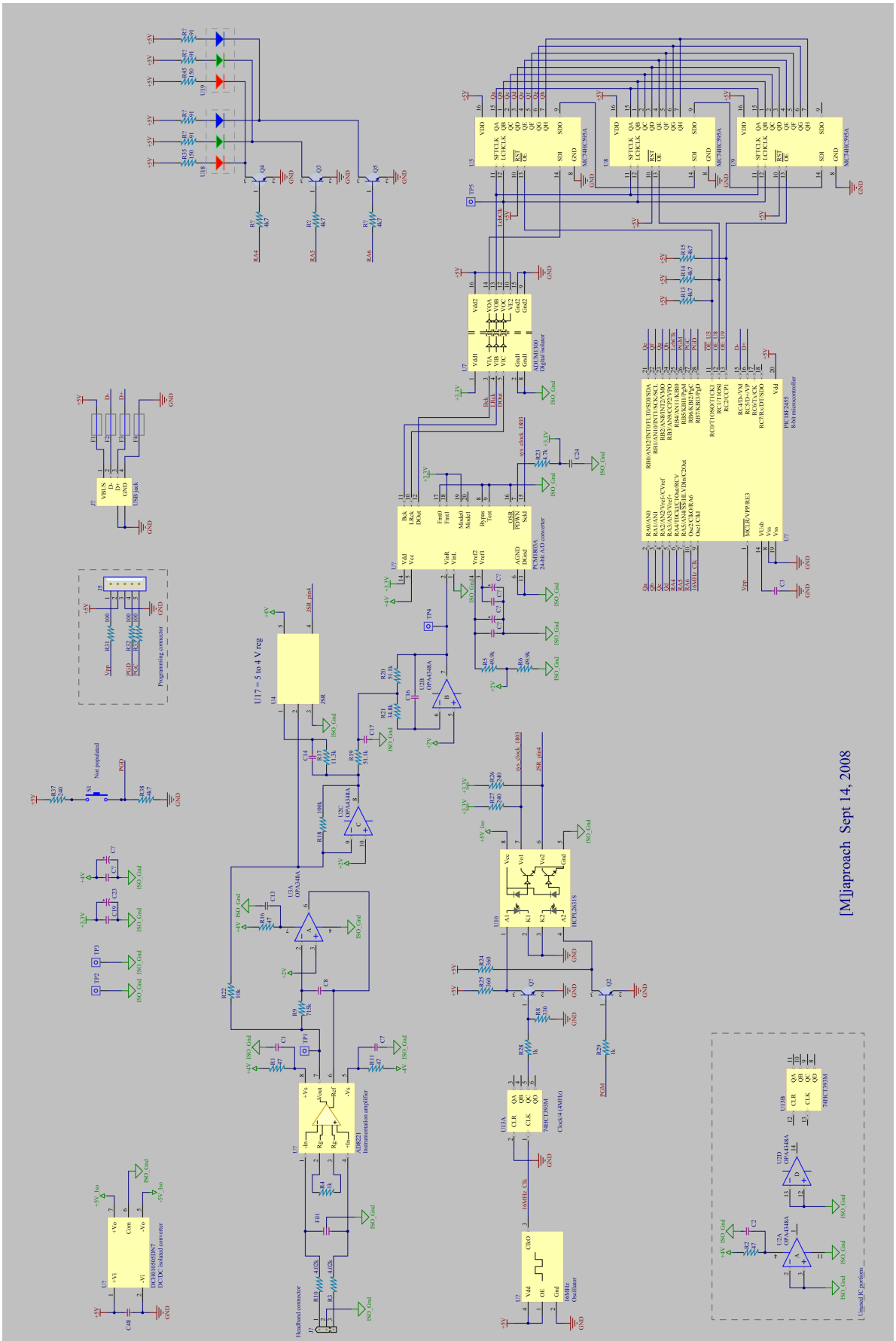

Abbildung A.1: Schaltplan des NIAs [`http://www.genmay.com/showthread.php?t=798717`, Stand Sept. 2009]

Quelltext A.1: `Interpret(Byte[] data)`

```
1   private void Interpret(Byte[] data)
2   {
3       int validPackets = data[55];//
4       peakMeterDataInPacket.SetData(new int[1] { (int)((double)(validPackets) /
            peakMeterDataInPacket.LEDCount * 100) }, 0, 1);
5
6
7       ulong packetTimer = (ulong)(data[54] * (1 << 8) + data[53] * (1 << 0) -
            validPackets);//NIA's timer is 16bit
8       if (chkRawSR.Checked)
9           ShowRawSamplingRate(validPackets);
10
11      /*
12       * the 56B-array holds the values (24-Bit-integer) of the read-out nia samples.
13       * byte-56 says how many samples are within that byte[].
14       *
15       */
16      for (ulong index = 0; index < (ulong)validPackets; index++)
17      {// go for all included/read samples
18
19          ulong timerPosition = (ulong)(packetTimer + index);
20          // construct one sample-value made of 3Bytes
21          long rawData = data[index * 3 + 1] * (1 << 0) //*2^0
22                       + data[index * 3 + 2] * (1 << 8) + //*2^8
23                       data[index * 3 + 3] * (1 << 16);//*2^16
24
25          _samplingQueue.Enqueue((float)rawData);
26
27          if (_samplingQueue.Count > 2048)
28              _samplingQueue.Dequeue();
29
30
31          ulong nextSamplingStep = (ulong)(_lastSamplingTime + _wantedSampleDuration);
32          ulong nextWantedSamplingTime = nextSamplingStep % _maxTimerValue;
33
34          long currSampleDurationOvertime = 0;
35
36          if (nextSamplingStep != nextWantedSamplingTime)//interval overflow
37          {
38              if (timerPosition >= _lastSamplingTime)//no overflow yet
39              {
40                  currSampleDurationOvertime = (long)(timerPosition - nextSamplingStep);
41              }
42              else
43                  currSampleDurationOvertime = (long)(timerPosition -
                        nextWantedSamplingTime);
44          }
45          else if (timerPosition < _lastSamplingTime)
46          {//timer overflow happened, but not interval
47              if (_maxTimerValue > nextSamplingStep)
48                  currSampleDurationOvertime = (long)(timerPosition +
                            (_maxTimerValue - nextSamplingStep));
49              else
50
```

```
51                      currSampleDurationOvertime = (long)_timerValue -
52                                (long)(nextSamplingStep - _maxTimerValue);
53              }
54              else
55                  currSampleDurationOvertime = (long)(timerPosition - nextWantedSamplingTime);
56
57              if (currSampleDurationOvertime < 0)//to counteract overflow-issues
58              {//still inside wanted interval
59                  _samplingCounter++;
60                  _samplingSum += rawData;
61              }
62              else
63              {//enough data collected
64                  double overtimeFactor = 0;
65                  if ((overtimeFactor =
66                        (Math.Abs((double)currSampleDurationOvertime / _wantedSampleDuration))
                            ) > 1)
67                  {
68                      nextSamplingStep = (ulong)(_lastSamplingTime +
69                                          ((ulong)overtimeFactor + 1) *
                                                _wantedSampleDuration);
70                      nextWantedSamplingTime = nextSamplingStep % _maxTimerValue;
71                  }
72
73                  timerPosition = _lastSamplingTime = nextWantedSamplingTime;
74
75                  //_lastTimer = timerPosition;
76                  DateTime currentLast5Second = DateTime.Now.AddSeconds(-5);
77                  //DateTime currentLastSecond = DateTime.Now.AddSeconds(-1);
78
79                 ...
80
81                  double originalRawData = _samplingSum > 0 ?
82                                           _samplingSum / _samplingCounter
83                                         : rawData;
84                  //process this averaged value
85                  _samplingCounter = 0;
86                  _samplingSum = 0;
87
88                  //number of left out zero-samples:
89                  long samplesToFill = (long)(nextWantedSamplingTime -
                         _lastOriginalRawDataTime)
90                                               / (long)_wantedSampleDuration;
91
92
93                  if (_lastSecond < currentLast5Second)//clear the zeroCounter,
94                  {
95                      _zeroSamplingCounterPerSecond = 0;//because it's been longer than 5
                            seconds,
96                      //when a zeroSample happened;
97                      _lastSecond = currentLast5Second;
98                  }
99
```

```
100                 _zeroSamplingCounterPerSecond += (int)samplesToFill - 1;
101
102                 ...
103
104                 while (samplesToFill-- > 0)
105                 {
106
107                     //calculate an average for the missing zero-samples interval(s)
108                     ulong step = (ulong)(samplesToFill) * _wantedSampleDuration;
109                     if (nextWantedSamplingTime > step)
110                         timerPosition = nextWantedSamplingTime - step;
111                     else
112                         timerPosition = _maxTimerValue - (step - nextWantedSamplingTime);
113
114                     rawData = (long)originalRawData//old dataValue
115                               - (long)((double)(nextWantedSamplingTime - timerPosition)//+ (
                                      wanted time
116                                   * (double)(originalRawData - _lastOriginalRawData)// *
                                         "DataSlope")
117                                  / (nextWantedSamplingTime - _lastOriginalRawDataTime)
                                        );
118
119                     ...
120
121                     //process that point:
122                     rawData = rawData >> 1; //max. 24b-int/2
123                     rawData = RawDataNormalization(rawData);
124                     rawData = RawDataPreProcessing((long)timerPosition, (long)rawData);
125
126                     rawData = FrequencyAnalysis((long)timerPosition, rawData);
127
128                     if (_timerValue > timerPosition + _wantedSampleDuration)
129                     {
130                         _timerValue = timerPosition;
131                         DataListBox.Items.Add("Timer Reset");
132                         UpdateZedGraph(0, 0);
133                     }
134                     _timerValue = timerPosition;
135
136                     string output = string.Concat("Timer: ", timerPosition, "\tData: ",
                            rawData);
137                     DataListBox.Items.Add(output);
138                     UpdateZedGraph(timerPosition, rawData);
139                 }
140                 _lastOriginalRawData = (long)originalRawData;
141                 _lastOriginalRawDataTime = timerPosition;
142
143
144             }
145         }
146     }
```