Institut für Visualisierung und Interaktive Systeme
Abteilung Mensch-Computer-Interaktion
Universität Stuttgart
Pfaffenwaldring 5a
D–70569 Stuttgart

Bachelorarbeit Nr. 11

Belastung als eine Eingabemodalität zur Interaktion mit graphischen Benutzungsoberflächen

Florian Straßer

Studiengang:	Informatik
Prüfer:	Prof. Dr. Albrecht Schmidt
Betreuer:	Dipl.-Inf. Bastian Pfleging
	Dipl.-Inf. Niels Henze
begonnen am:	21. Mai 2012
beendet am:	12. November 2012
CR-Klassifikation:	H.1.2, H.5.2, J.4, D.2, D.3

Kurzfassung

Graphische Benutzeroberflächen sind in unserem Alltag allgegenwärtig geworden. Unterschiede in der Benutzerfreundlichkeit werden deshalb auch sofort wahrgenommen und fallen als Differenzierungskriterium immer mehr ins Gewicht. In dieser Arbeit wird untersucht, ob Belastung von Nutzern als Eingabemodalität für graphische Benutzeroberflächen benutzt werden kann. Der Schwerpunkt in der Bestimmung der Belastung liegt dabei auf physiologischen Sensoren. Im ersten Schritt werden hierbei die für die Arbeit relevanten Hintergrundthemen betrachtet. Es werden verwandte Arbeiten eingeordnet und analysiert, sowie ein Konzept für einen adaptiven Prototyp einer Benutzeroberfläche entwickelt. Im Verlauf dieser Arbeit wird ein Programm entwickelt, das diverse physiologische Sensoren anbindet und die Sensordaten visualisiert und verarbeitet. Hierbei wird eine Echtzeit-Peak-Erkennung für das EKG- und BVP-Signal implementiert, mithilfe derer jeweils ein Echtzeit-Puls berechnet werden kann. Abschließend wird noch die Funktionalität der erstellten Software evaluiert.

Abstract

Graphical user interfaces have become ubiquitous in our daily lives. Differences in the ease of use are therefore perceived immediately and become more and more important as a differentiation criterion. This work will analyze if the workload a user is exposed to can be used as an input modality for a graphical user interface. The focus in calculating the amount of stress will be on physical sensors. In the first step, the relevant background knowledge for the work will be examined, related work will be classified and analyzed and a concept for a prototype of an adaptive user interface will be developed. In the course of this work, a program will be developed, that connects the various physiological sensors and visualizes the sensor data and processes. Here a real-time-peak recognition for the ECG- and BVP- signal will be implemented with the help of which a real time pulse can be calculated. The final part will evaluate the functionality of the developed software.

Inhaltsverzeichnis

Danksagung

Ich möchte allen Personen danken, die mich bei der Bearbeitung dieser Bachelorarbeit unterstützt und motiviert haben.

Zuerst einmal gilt mein Dank Professor Dr. Albrecht Schmidt, der sich immer wieder auch Zeit für Besprechungen, von der Themenfindung bis zum Abschluss der Arbeit, genommen hat und mir weitreichende Freiheiten in der Themenfindung eingeräumt hat. Auch möchte ich mich speziell für das entgegengebrachte Vertrauen bedanken, mir das Nexus10 für diese Arbeit auszuhändigen.

Des Weiteren möchte ich mich bei meinem Betreuer Bastian Pfleging für seine stete Unterstützung in allen Phasen der Arbeit bedanken, die bei Bedarf auch kurzfristige Meetings jederzeit einschloss. Meinem weiteren Betreuer Niels Henze danke ich für seine zahlreichen Korrekturhinweise bezüglich meiner Ausarbeitung.

Ein weiterer Dank geht an meinen Kommilitonen Marius Kleiner für die stets gute und freundschaftliche Zusammenarbeit. Auch meiner Freundin, gilt ein großer Dank, dafür dass sie mir einige Male entweder als Versuchsperson gedient oder mir in Selbstversuchen assistiert hat. Auch aufgrund ihrer Hilfe beim Korrekturlesen bin ich sehr dankbar. Ein besonderer Dank geht an meine Eltern die mich in meinem Studium stets moralisch, als auch finanziell zur Seite stehen und mich in allem stets herzlich unterstützen.

1 Einleitung

Graphische Benutzeroberflächen (engl. Graphical User Interface oder GUI) haben in unserem heutigen Privatleben und der Berufswelt in breitem Maße Einzug gehalten. Sie haben die Bedienung von Computern in vielerlei Hinsicht einfacher gemacht. Doch mittlerweile wird dieser Faktor durch immer komplexere Inhalte, die durch den technischen Fortschritt zwangsläufig entstehen und auch weiterhin entstehen werden, mehr als wettgemacht. Auch ist zu bemerken, dass das Interesse an Schulungen und Bedienungsanleitungen sinkt (Novick & Ward, 2006).

Um dieser Entwicklung entgegenwirken zu können, werden heutzutage auch graphische Benutzeroberflächen immer mehr auf Benutzerfreundlichkeit optimiert. Doch auch dort stößt man schnell an Grenzen, da die Anforderungen durch Benutzer mit verschiedenen Qualifizierungen, kulturellen Hintergründen und sogar auch verschiedenen temporären Befindlichkeiten von ein und demselben Benutzer stark auseinander gehen. Man stelle sich nur einmal einen Sicherheitsbeauftragten vor, der ein computergestütztes Sicherheitssystem einmal in entspanntem Zustand und ein anderes Mal während einer Gefahrensituation bedient. Die Auffassungsgabe und Lernfähigkeit sind in diesen beiden Szenarien vollkommen unterschiedlich. Dementsprechend verschieden sind auch die Anforderungen an eine Benutzeroberfläche. Im Fall eines gestressten Benutzers sollte beispielsweise die Oberfläche nicht noch weiteren Stress auslösen und sich auf das Wesentliche reduzieren. Einem unterforderten Anwender kann hingegen die gesamte Funktionalität mittels eines Klicks erreichbar gemacht werden. Somit kann eine Verbesserung der Benutzerfreundlichkeit im einen Fall eine Verschlechterung derselben im anderen Fall bewirken. Um auch dieses Problem zu umgehen ist eine mögliche Idee adaptive Benutzungsschnittstellen zu gestalten. Dies sind Benutzeroberflächen, die sich an den Benutzer und seine Bedürfnisse anpassen.

Die Frage ist, wie diese Anpassung / Adaptierung auf den Benutzer geschehen kann. Die einfachste Möglichkeit aus Sicht des Software-Entwicklers ist es, die Entscheidung der Art der Adaptierung dem Anwender zu überlassen. Die eleganteste Möglichkeit für den Anwender ist, dass dies das System vollkommen autonom, also ohne aktives Zutun des Benutzers, macht. Genau an diesem Punkt setzt diese Arbeit an. Ein Szenario, das den Zielgedanken der Forschung an dieser Thematik beschreibt, ist das folgende:

Student Tom betrachtet ein elektrisches Schaltbild an seinem Laptop. Da er das Bild nicht sofort versteht, erscheinen Beschriftungen an den Schaltsymbolen, die die Bedeutung jedes Schaltsymbols erklären. Auch nach Einblendung der Hinweise versteht Tom das Schaltbild jedoch immer noch nicht. Daraufhin ändert der Laptop die Darstellung des Schaltbildes und fasst Schaltsymbole zu Bauelementen zusammen. Mithilfe dieser

Darstellungsweise ist es dem Studenten möglich, die Schaltung zu verstehen. Das Computersystem erkennt auch diese Tatsache automatisch und lässt die Bauelemente wieder transparent werden.

1.1 Problemstellung

Diese Arbeit ist ein erster Schritt, die Belastung des Benutzers als eine Eingabemodalität für graphische Benutzeroberflächen zu nutzen. So wird ein neuer Weg in Richtung Benutzerfreundlichkeit und Benutzbarkeit von graphischen Benutzeroberflächen eröffnet. Um einen Überblick über das Fachgebiet zu erlangen, ist dazu zunächst die Erarbeitung relevanter Literatur notwendig. Dazu gehört auch die Erarbeitung von relevanter Literatur. Zur relevanten Literatur zählen sowohl Arbeiten zu Themen wie kognitive Last und Stress, als auch Arbeiten, die zumindest abschnittsweise ähnliche Dinge untersucht haben.

Für die Entwicklung eines adaptiven Systems ist es notwendig, geeignete Sensoren zu finden, mit deren Hilfe aussagekräftige Belastungswerte ausgelesen werden können. Daraufhin steht die Anbindung dieser Sensoren an. Wenn die Sensordaten verfügbar sind, müssen diese visualisiert werden und eventuell weiterverarbeitet werden. Ein Beispiel dafür ist die EKG-Kurve aus der der momentane Puls gewonnen werden kann. Dies alles geschieht immer im Hinblick darauf, dass aus den Sensordaten ein Belastungswert ableitbar sein soll, der als Eingabeparameter für Benutzeroberflächen dienen kann.

Sobald geeignete Sensoren ermittelt worden sind, kann im nächsten Schritt damit begonnen werden, einen Prototyp für eine adaptive Benutzungsschnittstelle zu entwickeln. Bei einer Überlastung des Nutzers wird der Informationsgehalt der aktuellen Darstellung gesenkt und / oder Hilfsinformationen anzeigt. Bei Unterforderung des Benutzers erfolgt das Gegenteil. Der Nutzen bzw. die Verbesserung der Benutzerfreundlichkeit dieses Programms kann wiederum im Rahmen einer Studie festgestellt werden.

Aufgrund des Umfangs der Problemstellung kann dies nicht alles in dieser Arbeit geleistet werden. Diese wurde jedoch bewusst so umfassend gewählt, um das Forschungsfeld zu explorieren. Praktisch kann in dieser Arbeit jedoch nur der erste Schritt umgesetzt werden, der stets auf das endgültige Ziel der Arbeit ausgerichtet ist.

In dieser Arbeit werden die Grundlagen und die verwandte Literatur, auch im Hinblick auf das Ziel einer adaptiven Benutzeroberfläche, untersucht. Danach werden geeignete Sensoren gefunden, die für die Bestimmung kognitiver Last geeignet sind. Im Anschluss daran erfolgt eine Anbindung der Sensoren an den Computer, um die physiologischen Daten des Anwenders abgreifen zu können. Für erste Analysezwecke wird ein

System zur Anzeige und Speicherung dieser Daten erstellt. Die Weiterverarbeitung der Datenströme zu aussagekräftigeren Werten kann hier ebenfalls geschehen.

1.2 Gliederung

Im folgenden Kapitel („Hintergrund“) werden einige für den weiteren Verlauf dieser Arbeit wichtigen Themen aufgegriffen und vertieft. Dabei wird beispielsweise der Begriff der Belastung näher erläutert, aber auch verschiedene technische Themen erklärt. Kapitel 3 stellt verwandte Arbeiten vor und gibt damit einen Überblick über relevante Arbeiten, die dieser Arbeit ähneln, oder Teilgebiete dieser Arbeit abdecken analysiert und klassifiziert. Ein Konzept für eine adaptive Benutzeroberfläche, wird in Kapitel 4 („Konzept“) vorgestellt. Danach werden im 5. Kapitel („Praktische Sensoranalyse“) verschiedene Sensortests durchgeführt. In Kapitel 6 („Implementierung“) wird die Anbindung der Sensoren bis hin zur Visualisierung und Verarbeitung der Sensordaten mit einer dafür entwickelten Software beschrieben. Im Anschluss daran wird in Kapitel 7 „Evaluierung“ diese Funktionalität mittels einiger Tests geprüft. Den Abschluss bildet das Kapitel „Zusammenfassung und Ausblick“, das die Ergebnisse der Bachelorarbeit zusammenfasst und einen Ausblick auf eventuell hierauf aufbauende Arbeiten gibt.

2 Hintergrund

In diesem Kapitel wird eine Wissensbasis geschaffen. Auf dieser Basis lässt sich der Rest der Arbeit besser verstehen. Es werden sowohl Begrifflichkeiten wie Belastung und Stress enger gefasst und erläutert, als auch allgemein wichtige Themen für diese Arbeit, wie die physiologischen Sensoren, erläutert.

2.1 Belastung

Ein zentraler Begriff in dieser Arbeit ist der Begriff der Belastung, wobei hier auf die psychische Belastung des Benutzers abgezielt wird. Belastung ist ein bewusst sehr breit gewählter Begriff, um das Forschungsgebiet möglichst allgemein zu untersuchen. Dieses reicht über körperliche Belastung bis hin zu Stress und kognitiver Last. Dennoch muss die Belastungen ein wenig auf die Belastungsarten eingeschränkt werden, die für diese Arbeit relevant sind. Nun ist zu klären: Welche Belastungsarten treten bei der Benutzung von graphischen Benutzeroberflächen auf?

2.1.1 Körperliche Belastung

Körperliche Belastung ist die physische Belastung, der der Körper ausgesetzt ist. Diese Belastungsart spielt bei der Bedienung von graphischen Benutzeroberflächen eher eine untergeordnete Rolle. Eine Ausnahme bilden hier neuerdings Eingabegeräte für Spielekonsolen. Beispiele hierfür sind die Nintendo Wii[1], die Kinect[2] oder die Play Station Move[3]. Jedoch ist die dort erzielte körperliche Belastung kein Maß dafür, ob der Benutzer mit der Bedienoberfläche überfordert ist. Deshalb wird die körperliche Belastung in dieser Arbeit auch nicht weiter vertieft werden.

Trotzdem kann die Erfassung von Biofeedback-Werten auch hier interessant sein. Ein Beispiel hierfür kann z.B. der Puls sein, mit dessen Hilfe man die Trainingsintensität auf den Benutzer anpassen, oder auch das Spiel bei gefährlichen Werten pausieren könnte. Dies ist insbesondere durch eine kürzlich erst veröffentlichte Methode der Videoverarbeitung, die am Massachusetts Institute of Technology (MIT) veröffentlicht wurde, interessant geworden. Die Methode nennt sich „Eulersche Videoverstärkung“ (Wu, et al., 2012) und macht Dinge wie Atmung und Puls in einem Videobild sichtbar. Dies könnte also beispielsweise mit einer entsprechend angepassten Kinect-Konsole durchaus ohne weitere Geräte und kontaktlos realisiert werden.

[1] http://www.nintendo.de/

[2] http://www.xbox.com

[3] http://www.playstation.com/

2.1.2 Stress

Eine weitere Belastungsart ist Stress. Stress ist als eine Reaktion des Organismus auf verschiedene Stressfaktoren definiert, die sich z.B. in der vermehrten Ausschüttung von Stresshormonen äußert. Stressfaktoren können hierbei sowohl körperliche als auch seelische Belastungen sein. Typische Stressfaktoren (auch Stressoren genannt) sind z.B. (Walter de Gruyter GmbH & Co. KG, 2010, p. 1991):

- Krankheit
- Trauma
- Umwelteinflüsse
- emotionale Belastung

Die oben genannten Stressfaktoren sind allerdings eher als Hintergrundfaktoren bei der Benutzung von graphischen Benutzeroberflächen anzusehen, beispielsweise also eher der Druck von außen auf den Benutzer wenn etwas nicht funktioniert, als die Probleme selbst, die bei der Benutzung der graphischen Benutzeroberfläche auftreten.

Auch diesen Problemen muss begegnet werden um eine fehlerfreie Bedienung der Software auch in solchen Situationen zu ermöglichen. Die Frage ist nur, ob dies die Benutzeroberfläche alleine bewerkstelligen kann. Ein Beispiel, das dagegen spricht, wäre beispielsweise folgendes Szenario:

Der Mitarbeiter Tom wird zu seinem Chef ins Büro zitiert, wo ihm mächtig Druck gemacht wird bezüglich der Präsentation für das Meeting in 3 Stunden. Der Chef trägt ihm eine Menge Aufgaben auf, wie die Präsentation noch verbessert werden soll, und möchte schon in einer Stunde Ergebnisse sehen.

In dieser Situation würde es Tom wenig nützen, wenn die Software ihm aufgrund seines Stresslevels Hilfetexte einblendet, da er ja schnell arbeiten muss. Diese würden also eher stören. Auch eine Verringerung der angezeigten Programmfeatures würde sich eher negativ auf Tom's Leistungsfähigkeit auswirken, da er evtl. gerade diese Programmfeatures benutzen muss um schnell die geforderten Ergebnisse zu erzielen.

Jedoch kann eine Adaptierung gemäß des Stresslevels auch durchaus sinnvoll, wie im EU-Projekt NIFTi[4] (Natural Human-Robot Interaction in Dynamic Environments) deutlich wird. Bei diesem Projekt geht es darum Roboter zu entwickeln, die in Zukunft Rettungskräfte in gefährlichen Einsatzgebieten unterstützen sollen. Dabei soll es möglich sein, dass die menschlichen Rettungskräfte durch natürliche Sprache mit den Robotern kommunizieren können. (Surmann & Worst, 2012)

[4] http://www.nifti.eu/

Ziel ist es diese Roboter intelligent genug zu machen, um die Gemütslage ihrer Befehlsgeber zu erkennen. Falls erkannt wird, dass der Befehlsgeber gerade unter großem Druck und damit unter Stress steht, reagiert der Roboter entsprechend. Bei scheinbar klaren Anweisungen, stellt der Roboter Rückfragen, da es gut sein kann, dass der befehlsgebende Mensch Informationen über den Kontext voraussetzt, die die Maschine nicht hat. In mehreren Feldtests wurden Parameter wie Herzfrequenz, Gesichtsausdruck und die Stimme von Rettungskräften aufgezeichnet, um Merkmale für Stress zu finden. (Fleschner, 2012)

2.1.3 Kognitive Last

Kognition ist die Sammelbezeichnung für alle Prozesse, die mit dem Erkennen zusammenhängen, z.B. (Walter de Gruyter GmbH & Co. KG, 2010, p. 1084):

- Vorstellung
- Gedächtnis
- Lernen / Aufmerksamkeit
- Erinnerung

In (Oviatt, 2006) wird kognitive Last im Zusammenhang mit graphischen Benutzeroberflächen betrachtet. Aufmerksamkeit und das Gedächtnis gelten als der Flaschenhals der menschlichen Informationsverarbeitung und somit auch die Arbeit mit graphischen Benutzeroberflächen. Um kognitive Last in einem Experiment zu erzeugen, ist die gebräuchlichste Form eine sogenannte „divided attention“ bzw. „dual-task“ Aufgabe. Dies sind Aufgaben, in denen der Benutzer zwei oder mehrere Dinge parallel tun muss, meist in der Form, dass es eine primäre Aufgabe gibt, die die meiste Zeit in Anspruch nimmt, dazu parallel jedoch noch eine sekundäre Aufgabe durchgeführt werden muss, die immer wieder kurze Zeitabschnitte in Anspruch nimmt.

2.1.3.1 Cognitive Load Theory

In der Quelle „Cognitive Architecture and Instructional Design“ (Sweller, et al., 1998) wird unter anderem ein Einblick in die Cognitive Load Theory gegeben. Die Cognitive Load Theory (CLT) befasst sich mit der beim Lernen verursachten kognitiven Last. Dabei wird vor allem die beim Lernen verursachte kognitive Last etwas feingranularer eingeteilt. Damit hilft die CLT dabei, Lern- bzw. Arbeitsbedingungen so zu gestalten, dass diese dem Benutzer nicht unnötige Schwierigkeiten bereiten, die über die Schwierigkeiten der zu verstehenden Materie an sich hinausgehen.

Die CLT beschreibt drei verschiedene Arten der kognitiven Belastung beim Lernen (Sweller, et al., 1998):

- **Expertise / Intrinsic Cognitive Load:** Diese Art der kognitiven Last wird durch den zu lernenden Stoff selbst verursacht. Intrinsic Cognitive Load beschreibt al-

so die Menge an Information, die nach dem Lernprozess im Langzeitgedächtnis gespeichert wird.

- **Extraneous Cognitive Load:** Bei dieser Art wird die bei Expertise ausgesparte Darstellung des Stoffes für sich allein betrachtet. Es zählt also nur die Aufmachung des Stoffes. Jedoch sollte natürlich eine gute Aufmachung des Stoffes immer zum Stoff selbst passen. Auch die Aufmachung sollte natürlich auf den Benutzer zugschnitten sein. Man stelle sich nur beispielsweise eine Darstellung vor, die in einer Sprache geschrieben ist, die der Benutzer nicht versteht.
- **Germane Cognitive Load:** Hier geht es um die kognitive Last die für den reinen Lernprozess benötigt wird. Also wird hierbei rein die Schwierigkeit beschrieben den zu lernenden Stoff zu verstehen und alle Verbindungen herstellen zu können.

Die konkreten Belastungen der oben genannten Belastungsarten lassen sich zusammen addieren. Wenn die Summe eine zu hohe kognitive Last für den Benutzer bedeutet, wird es schwer sein, für den Benutzer den Stoff auf diesem Wege zu lernen. (Sweller, et al., 1998)

In vielen Fällen lassen sich Intrinsic Cognitive Load und Germane Cognitive Load nicht beeinflussen. Ist also die Gesamtlast zu hoch, bleibt nur die Möglichkeit, den Extraneous Cognitive Load zu veringern. Dafür gibt es eine Vielzahl an weitgehend allgemeingültigen Rezepten, jedoch hängt eine gute Darstellung auch sehr stark vom Benutzer ab. So spielt es bei der Entscheidung, eine passende Darstellung zu wählen, eine fundamentale Rolle über welches Vorwissen der Benutzer verfügt. (Sweller, et al., 1998)

2.1.3.2 Bedeutung für diese Arbeit

Zusammenfassend lässt sich sagen, dass der Level an kognitiver Last, den Level an Überlastung eines Nutzers an einer Benutzeroberfläche, im Vergleich zu Körperlicher Belastung und Stress, am besten wiedergibt. Natürlich steigt und fällt die kognitive Leistungsfähigkeit auch mit dem Stresslevel des Benutzers, weshalb der Stress nie vollständig beiseitegelassen werden kann. Dies ist aber genau aus diesem Grund auch gar nicht erwünscht.

Wenn wir einmal annehmen, dass die dargebotenen Sachverhalte im Grunde von dem Nutzer verstanden und verinnerlicht werden können, wissen wir, dass das Problem ausschließlich in der Darstellung liegt und dass wir auch nur diese verändern müssen. Also ist die Frage, wie eine solche kognitive Überlastung des Benutzers erkannt werden kann.

2.2 Physiologische Sensorik

Um die zuvor beschriebene kognitive Überlastung zu erkennen, ist der Ansatz dieser Arbeit die Auswertung verschiedener physiologischer Sensoren um deren Zusammenhänge mit psychologischen Faktoren zu erkennen und zu nutzen.

2.2.1 EKG-Messung

Ein EKG-Signal zeigt die elektrischen Aktivitäten des Herzens, die elektrischen Signale bewegen die Herzmuskelfasern zum zyklischen Kontrahieren, was zum Pumpen des Blutes führt. Im EKG-Signal ist zu sehen, wie schnell das Herz schlägt, ob das Herz regelmäßig schlägt und bei etwas detaillierterer Betrachtung die Funktionalität der verschiedenen Teile es Herzens. Für den letzten Punkt ist allerdings eine umfangreiche Auswertung der Artefakte im EKG-Signal, als auch oft die Durchführung eines Mehrkanal-EKGs nötig. Diese Artefakte bestehen im Wesentlichen aus den verschiedenen Spitzen (Peaks) und den Intervallen dazwischen. (National Heart Lung and Blood Institute, 2010)

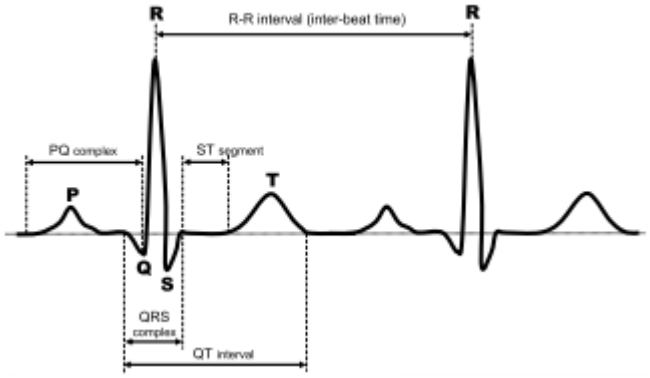

Abbildung 1: Eine EKG-Kurve mit eingezeichneten P-, Q-, R-, S- und T-Peaks und den Intervallen zwischen diesen Peaks. (Riener, et al., 2009)

Wesentlich für diese Arbeit, ist der Puls bzw. das R-R-Intervall (Zeit zwischen zwei R-Peak-Spitzen im EKG, siehe dazu Abbildung 1). Um den Puls zu bestimmen, müssen deshalb diese R-Peaks erkannt werden, um das R-R-Intervall berechnen zu können. Eine Umrechnung ist mit folgender Formel möglich:

$$Herzfrequenz = \frac{60\ Schläge}{(inter - beat\ interval)}$$

Formel 1: Umrechnungsformel inter-beat Intervall in Herzfrequenz

2.2.2 Hautleitwertmessung

Der Hautleitwert (engl. Galvanic Skin Response - GSR) wird meist am Finger oder der Handfläche gemessen. Zur Messung wird zwischen zwei Auflagepunkten eine kleine Spannung geleitet und der Spannungsabfall zwischen den Elektroden gemessen. Daraus

kann der Widerstand bzw. der Leitwert, der Kehrwert des Widerstands, berechnet werden, den die Haut zwischen den beiden Punkten besitzt. Wesentlichen Einfluss auf den Hautleitwert hat die Feuchtigkeit der Haut, die durch Schwitzen hervorgerufen wird, dies führt zu einem Anstieg des Hautleitwertes. Das Schwitzen an den Händen wird neben körperlicher Belastung auch stark durch Stress und kognitive Last beeinflusst, so führt Stress zu einem erhöhten Hautleitwert. (Sung & Pentland, 2005)

2.2.3 Hauttemperaturmessung

Die Temperatur ist eine der natürlichsten Formen die Befindlichkeit eines Menschen zu überprüfen. Man denke nur einmal an eine Mutter, die bei ihrem Baby die Körpertemperatur durch Handauflage an der Stirn überprüft, sobald sie das Gefühl hat, dass es dem Kind nicht gut geht. Die bekanntesten Einflüsse auf die Körpertemperatur sind Krankheiten in Verbindung mit Fieber. Die Körpertemperatur steigt jedoch auch bei jeglicher Beanspruchung des Körpers, die zu einem erhöhten Energieverbrauch führt. Solche Faktoren der Beanspruchung können physisch, psychisch, oder auch nur ein erhöhter Stoffwechsel und Energieverbrauch aus andern Gründen sein. (Sung & Pentland, 2005)

Der Hauttemperatursensor geht einen Schritt von der reinen Körpertemperatur und damit auch von dem großen Einfluss physischer Aktivität weg. Bei entsprechenden Sensoren handelt es sich um einen Temperatursensor der an entsprechenden Körperstellen angebracht wird. Geeignete Körperstellen sind oft die Finger, die Zehen oder auch die Stirn. Für die Temperaturwerte spielt hier die Durchblutung der Körperteile, die nahe am Sensor liegen eine entscheidende Rolle. Der Blutfluss wird durch physischen Stress beeinflusst, da sich hierbei Blutgefäße verengen, was eine geringere Durchblutung bewirkt (Mind Media B. V., 2006). So gibt es einen Zusammenhang zwischen abfallender Hauttemperatur und Stress.

Ein großes Problem ist, dass die Hauttemperatur nicht isoliert betrachtet werden kann, sondern dass auch die Umgebungstemperatur einen großen Einfluss hat. Dieser Einfluss erscheint zunächst einmal eher unwichtig, geht man davon aus, dass die Umgebungstemperatur konstant bleibt. Jedoch ändern bereits minimalste Bewegungen die Beeinflussungsverhältnisse beträchtlich.

2.2.4 Blutvolumenpulsmessung

Der Blutvolumenpuls (BVP) wird meist an einem Finger gemessen, kann aber auch an einem Zeh gemessen werden. Ein Blutvolumensensor misst den relativen Blutfluss, durch Verwendung einer optischen Elektronik. Bei jedem Herzschlag wird Blut durch Arterien und Blutgefäße gepumpt an der Spitze des Blutflusses durch die Körperstelle an der der Sensor angebracht ist, erreicht auch das BVP-Signal seinen höchsten Wert.

Ein wesentlicher Vorteil des BVP ist, dass das Anlegen des Sensors wesentlich einfacher ist, als beim EKG. (Mind Media B. V., 2006)

2.2.5 Muskelaktivität des Trapezmuskels

Der Trapezmuskel hat als wichtigste Aufgabe das Verschieben und Drehen des Schulterblatts. Der Trapezmuskel gehört zur Muskelgruppe der Schultergürtelmuskulatur und liegt damit an der Rückseite des Körpers angrenzend an die Halsregion, wie in Abbildung 2 zu sehen. (Wittkowski & Speckmann, 2006, p. 169)

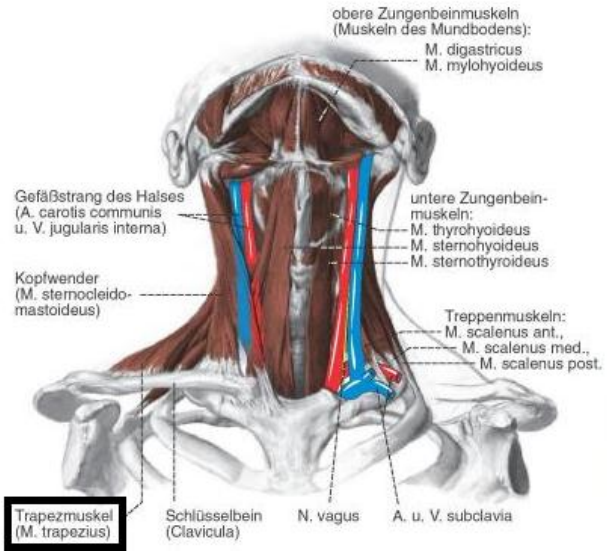

Abbildung 2: Rückenmuskulatur mit eingezeichnetem Trapezmuskel (Wittkowski & Speckmann, 2006)

2.3 Java

Da der Programmierteil dieser Arbeit weitgehend mit Java durchgeführt wird, wird hier eine kurze Einführung in die Sprache Java gegeben.

Java ist eine objektorientierte Programmiersprache, die seit 1991 von Sun Microsystems[5] entwickelt wurde (zunächst unter dem Namen „Oak“). Die Syntax basiert weitgehend auf C bzw. C++. Java ist eine platformunabhängige Sprache, also ist es möglich ein und denselben Programmcode inklusive der GUI (Graphical User Interface) auf unterschiedlichen Plattformen auszuführen, ohne dass Anpassungen nötig sind. (Prof. Dr. Volker Claus, 2006, p. 329)

Unterstützt werden übliche elementare Datentypen, wie `boolean`, `integer`, `float`. Weitere Datentypen (und auch die elementaren Datentypen) werden durch

[5] Heute Oracle http://www.oracle.com

Klassen bereitgestellt. Weiterhin gibt es übliche Kontrollstrukturen wie `if` oder `switch` und Schleifenstrukturen wie `while` oder `for`.

Weiterhin gibt es Klassen, die sowohl Attribute als auch Prozeduren enthalten können. Wie bei den Datentypen schon angedeutet, ist in Java nahezu alles eine Klasse. So ist ein Java Programm auch als eine Folge von Klassen definiert, wobei es für ein Programm auch nur eine einzige main-Methode geben darf. Sie ist der Einsprungspunkt beim Starten des Programms. Die main-Methode befindet sich dabei in einer der Klassen des Programms. Auch Klassen können zu übergeordneten Programmeinheiten, den Paketen (engl. packages), zusammengefasst werden. Mithilfe von geschweiften Klammern lassen sich zusammenhängende Teile im Programm definieren, die sogenannten Blöcke. Von außerhalb ist der Inhalt eines Blockes „nicht sichtbar“, was bedeutet, dass auf darin definierte Variablen und Methoden nicht zugegriffen werden kann. (Prof. Dr. Volker Claus, 2006, pp. 330-331)

Das erste Programm beim Erlernen einer neuen Programmiersprache ist für gewöhnlich das „Hallo-Welt-Programm“ (engl. Hello World), dieses kann in Java beispielsweise aussehen wie in Listing 1 dargestellt.

```
1     public class HelloWorld
2     {
3           public static void main(String[] args)
4           {
5                 System.out.println("HelloWorld");
6           }
7     }
```

Listing 1: Hallo-Welt-Programm in Java

Mit Java lassen sich grundsätzlich zwei Arten von Programmen entwickeln. Zum einen gibt es die Möglichkeit Applikationen zu erstellen. Dies sind selbstständige, direkt vom Java-Interpreter ausgeführte Programme. Die andere Art sind sogenannte Applets, diese werden in HTML-Seiten eingebunden. (Prof. Dr. Volker Claus, 2006, p. 331)

Java-Quellcode wird vom Compiler in plattformunabhängigen Bytecode übersetzt. Dieser platformunabhängige Bytecode wird auf dem Zielsystem in Maschinenanweisungen umgewandelt, dies macht ein Interpreter. Der Interpreter für diesen Bytecode ist bei Java die Java Virtual Machine (JVM). (Prof. Dr. Volker Claus, 2006, p. 331)

Java unterstützt alle wichtigen Konzepte der Objektorientierung, wie Klassen, Interfaces, die Definition von Methoden sowohl auf Instanzebene als auch auf Klassenebene, Vererbung (bei Interfaces auch Mehrfachvererbung) und dynamisches Binden (also das Binden der Klassen zur Ausführungszeit und nicht schon zur Compilezeit). Auch Konzepte zur Parallelität werden unterstützt, wozu z.B. Threads und deren Synchronisation mittels Sperren gehören. Auf fehleranfällige Sprachkonstrukte wurde in Java weitgehend verzichtet. So unterstützt Java keine Konstrukte wie Zeigerarithmetik, implizite

Typkonvertierung oder Präprozessoranweisungen. So sind viel mehr Prüfungen bereits zur Übersetzungszeit möglich. Jedoch können Dinge wie die Prüfung auf Nullzeiger oder der Indexgrenzen von Arrays auch bei Java nur zur Laufzeit geprüft werden. (Prof. Dr. Volker Claus, 2006, pp. 331-332)

Java verwendet eine implizite Speicherallokation, d.h. Speicherplatz muss nicht freigegeben werden, wenn eine Variable nicht mehr benötigt wird. Diese Aufgabe übernimmt ein Garbage Collector, dieser gibt den Speicher einer Variablen automatisch frei, sobald keine Referenz mehr darauf im System existiert. (Prof. Dr. Volker Claus, 2006, pp. 331-332)

2.4 Kommunikationsprotokolle

In diesem Kapitel werden die für diese Arbeit relevanten Kommunikationsprotokolle erklärt. Dies ist zum einen der Funkstandard Bluetooth und zum anderen das Netzwerktransportprotokoll Transmission Control Protocol (TCP).

2.4.1 Bluetooth

Bluetooth ist ein internationaler Standard für ein Funknetz mit einer Reichweite, die bei durchschnittlicher Leistung der Geräte bei ca. 10-100 Metern liegt. Der Name stammt von einem dänischen König im 1. Jahrtausend mit dem Namen Harald Blåtand, engl. Harold Bluetooth. Bluetooth wurde 1994 von der Firma Ericsson entwickelt, gedacht ist es als eine Art Kabelersatz für Kleingeräte wie Handys, Tastaturen, Computermäuse oder Kopfhörer. (Prof. Dr. Volker Claus, 2006, p. 107)

Primäres Entwicklungsziel war, wie bei den meisten WPANs (wireless personal area networks) ein möglichst niedriger Energieverbrauch. Ein weiteres Entwicklungsziel war eine ausreichend hohe Übertragungsrate, damit auch Multimediaanwendungen möglich sind. Ein weiterer Punkt war ein möglichst niedriger Preis um es zu ermöglichen Bluetooth auch in niederpreisigen Geräten zu integrieren. Des Weiteren lag der Fokus auf der automatischen Kopplung von Geräten, wobei hier gewisse Sicherheitsstandards zu beachten waren. Deshalb müssen Geräte die erstmalig eine Verbindung zueinander aufbauen sollen zunächst durch den Benutzer dazu autorisiert werden um, wie bereits erwähnt, künftige Kopplungen automatisch durchführen zu können. (Prof. Dr. Volker Claus, 2006, pp. 107-108)

Der Frequenzbereich von Bluetooth liegt um 2,4 Gigahertz. Ein Bluetooth Netz kann bis zu 255 Geräte umfassen. Bluetooth liegt mittlerweile schon in der Version 4.0 vor, welche jedoch abwärtskompatibel zu allen Vorgängerversionen ist. Somit muss bei der Kopplung von Geräten nicht auf die Version des Bluetooth-Moduls geachtet werden. (Prof. Dr. Volker Claus, 2006, pp. 107-108)

Als Konkurrent von Bluetooth hat sich ZigBee, ein von Philipps entwickelter Standard für ein WPAN, entwickelt, welches den Frequenzbereich von 2,4 Gigahertz nutzt. ZigBee hat zwar eine geringere Datenübertragungsrate als Bluetooth, aber auch einen geringeren Stromverbrauch. (Prof. Dr. Volker Claus, 2006, pp. 107-108, 746-747)

2.4.2 TCP

Transmission Control Protocol (TCP) ist im Gegensatz zum User Datagram Protocol (UDP) ein Übertragungsprotokoll, das die Übertragung eines zuverlässigen Bytestroms zwischen zwei Punkten, die über ein unzuverlässiges Netzwerk verbunden sein können, garantiert. Dabei wird sowohl sichergestellt, dass versandte Pakete beim Empfänger ankommen, als auch, dass diese in der versandten Reihenfolge ankommen. TCP simuliert somit einen zuverlässigen Kommunikationskanal. (Tanenbaum, 2003, pp. 580-581)

TCP liegt im siebenschichtigen ISO-OSI-Referenzmodell (Abbildung 3: ISO-OSI-Referenzmodell) in der Transport-Schicht, der vierten Schicht. Kann also je nachdem welche Protokolle in den Schichten darunter verwendet werden schon bestimmte Dinge voraussetzen. Beispielsweise kann jeder Host in endlicher Zeit erreicht werden. Oder auch, dass das Routing in der Netzwerkschicht geschieht.

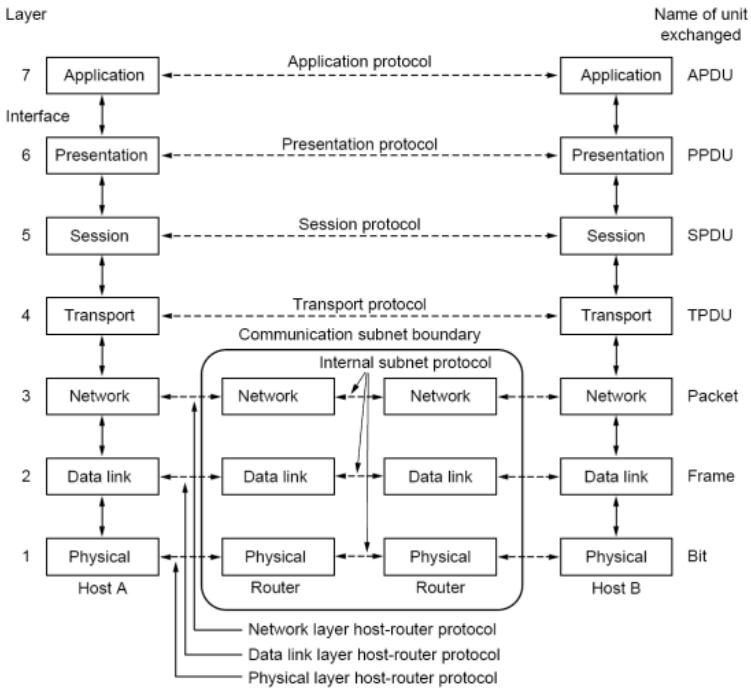

Abbildung 3: ISO-OSI-Referenzmodell (Tanenbaum, 2003, p. 56)

Da TCP meist in Verbindung mit IP (Internet Protocol) (auch TCP/IP genannt) verwendet wird, muss TCP mit den Möglichkeiten des IP-Protokolls die oben genannten Eigenschaften herstellen. Da die IP-Schicht keinerlei Gewähr bietet, ob gesendete Datagramme ankommen, muss TCP mithilfe eines Timeout-Intervalls gewährleisten, dass

Pakete gegebenenfalls erneut gesendet werden. Deshalb benötigt TCP auch Sequenznummern.

Der TCP-Header enthält unter anderem folgende Felder:

- Quellport (2 Byte)
- Zielport (2 Byte)
- Folgenummer (Sequenznummer des ersten Daten-Bytes) (4 Byte)
- Bestätigungsnummer (Sequenznummer, die als nächstes erwartet wird) (4 Byte)
- Prüfsumme (2 Byte)
- Optionen (variable Länge)

Daran anschließend folgen die Daten aus Sicht der Transportschicht. Diese können aus Sicht der darüber liegenden Schichten allerdings wiederum auch Headerinformationen sein. (Tanenbaum, 2003, pp. 585-587)

3 Verwandte Arbeiten

In diesem Kapitel wird ein Überblick über die verwandte Literatur zu dieser Arbeit gegeben werden, also Artikel, in denen es um die Bestimmung von Arbeitslast vorzugsweise durch physiologische Sensoren geht. Dabei wurden sowohl Artikel untersucht, bei denen es um kognitive Last als auch solche, bei denen es um Stress geht, da die dabei verwendeten Methoden oft ähnlich sind. Des Weiteren soll ein Einblick in alternative Wege aufgezeichnet werden, Arbeitslast ausschließlich über physiologische Sensoren zu bestimmen.

3.1 Studien zur Bestimmung der Belastung

Um die Arbeitsbelastung in einer Studie zu untersuchen, müssen geeignete Aufgaben für die Studie gefunden werden. Diese Aufgabenstellungen müssen definiert verschiedene Level an kognitiver Last bei den Probanden hervorrufen, um aus den Sensordaten einen Index für die Arbeitsbelastung erzeugen zu können. Dazu müssen erst die Zusammenhänge zwischen Sensorwerten und der momentanen Arbeitslast hergestellt werden. Die verschiedenen Möglichkeiten solche Studien zu gestalten werden in diesem Abschnitt diskutiert.

Eine Überforderung wird in den Studien zum einen durch die vom Probanden vorgeschlagene Lösung und zum anderen durch die für die Lösung gebrauchte Zeit festgestellt. Das Erkennen einer Unterforderung ist wenn überhaupt nur über die benötigte Zeit möglich. Eine weitere Möglichkeit ist die Selbsteinschätzung der Probanden zu nutzen. Weiterhin sollte es möglich sein, die Aufgaben in der Schwierigkeit zu variieren, um den Schwierigkeitsgrad im Verlauf der Studie zu steigern, oder auch um dem unterschiedlichen Ausgangsniveau der Probanden, die an der Studie teilnehmen, gerecht zu werden.

Meist werden mehrere der im Folgenden aufgeführten Versuchsarten in einer Studie verwendet. Wobei der Grund weniger darin liegt, dass eine einzelne Methode nicht aussagekräftig genug wäre, sondern dass aus ein Probanden so mehr Daten liefert und somit die Studie weniger aufwändig wird.

3.1.1 Aufgaben zur Bilderkennung

In (Haapalainen, et al., 2010) wird ein Studienaufbau verwendet in dem unvollständige Strichzeichnungen verwendet werden, um verschiedene Level kognitiver Belastung zu erzeugen.

Bei dieser Aufgabenstellung wird eine schemenhafte Strichzeichnung verwendet. Von dieser werden entweder viele kleine oder wenige große Teile entfernt. Daraufhin muss die Versuchsperson versuchen zu erkennen, um was es sich in dem Bild handelt.

Abbildung 4 zeigt eine solche Strichzeichnung, die einen Bogenschützen darstellt. Darunter befindet sich ein Eingabefeld, in das die Lösung eingegeben werden muss. Andere Lösungen zur Eingabe wären entweder mehrere Auswahlmöglichkeiten anzubieten oder eine Spracheingabe, wobei im Rahmen einer Studie nur eine „Wizard of Oz“ Implementierung der Spracheingabe sinnvoll wäre. (Haapalainen, et al., 2010)

Abbildung 4: Eine mögliche Aufgabe einer Studie zur Bestimmung der Belastung mittels Bilderkennung. Abgebildet ist eine lückenhafte Strichzeichnung eines Bogenschützen. (Haapalainen, et al., 2010)

3.1.2 Aufgaben zum Mustervergleich

Beim Mustervergleich ist eine musterhafte Strichzeichnung (Modell) gegeben und daneben sind wiederum mehrere solche Strichzeichnungen abgebildet. Die Aufgabe besteht darin herauszufinden in welchen Bildern das erste Modell enthalten ist. Die Lösung kann der Proband beispielsweise über Checkboxen unter den Bildern markieren. Abbildung 5 zeigt eine solche Aufgabenstellung inklusive der bereits korrekt markierten Lösung. (Haapalainen, et al., 2010)

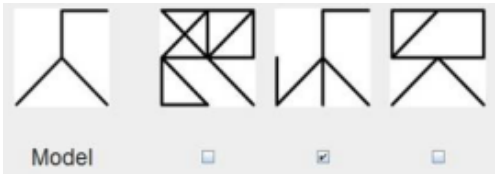

Abbildung 5: Eine mögliche Aufgabe einer Studie zur Bestimmung der Belastung mittels Mustervergleich. Beispiel einer Musterwiedererkennung bestehend aus einem Modell und drei weiteren Abbildungen, in denen das Modell einmal wiederzufinden ist. (Haapalainen, et al., 2010)

Eine Steigerung der Schwierigkeit der Aufgabe ist hier über die Änderung der Komplexität der einzelnen Bilder möglich. Z.B. durch Erhöhung der Linienanzahl sowohl vom Modell als auch der Vergleichsbilder. (Haapalainen, et al., 2010)

3.1.3 Aufgaben zum Suchen von Buchstaben in Wörtern

Hierbei wird dem Versuchsteilnehmer eine Reihe von Wörtern gezeigt mit der Aufgabe, alle Wörter zu markieren, in denen ein bestimmter Buchstabe vorkommt, wobei hier

meist der Buchstabe „A“ verwendet wird. Die Versuchsperson bekommt dabei bereits vorgegeben, wie viele Wörter aus der Liste ein „A“ enthalten. Sie soll alle dieses Wörter finden und markieren. (Haapalainen, et al., 2010)

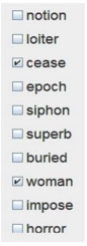

Abbildung 6: Eine mögliche Aufgabe einer Studie zur Bestimmung der Belastung mittels der Aufgabe Buchstaben in Wörtern zu suchen. Finden von dem Buchstaben „A“ in einer Reihe von Wörtern (Haapalainen, et al., 2010)

In Abbildung 6 ist eine solche Aufgabe inklusive Lösung dargestellt. Allerdings umfasste die Wortliste bei der durchgeführten Studie 40 Wörter. Dabei gab es zwei Schwierigkeitsstufen: In der leichteren Variante bestanden die Wörter aus 3-5 Buchstaben, in der schwereren Variante waren es 7-9 Buchstaben. (Haapalainen, et al., 2010)

Eine Erhöhung der Komplexität der Aufgabe lässt sich über die Erhöhung der Länge der Wörter erreichen. Dabei ist darauf zu achten, entweder Wörter mit wenigen A’s zu verwenden oder zu einem weniger häufigen Buchstaben zu wechseln, da sonst meist nicht das gesamte Wort angesehen werden muss, sondern nur der Anfang oder das Ende. (Haapalainen, et al., 2010)

3.1.4 Aufgaben zum Zahlenvergleich

Bei dieser Aufgabe werden den Teilnehmern Zahlenpaare gezeigt von nahezu identischen oder identischen mehrstelligen Zahlen. Wenn die Zahlen sich unterscheiden, soll dieses Zahlenpaar markiert werden, dies kann wiederum per Checkbox passieren. (Haapalainen, et al., 2010)

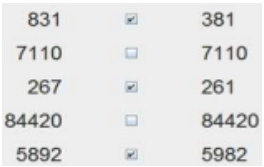

Abbildung 7: Eine mögliche Aufgabe einer Studie zur Bestimmung der Belastung mittels Zahlenvergleich. Abgebildet ist eine Aufgabe des Zahlenvergleichs von mehreren mehrstelligen Zahlen. (Haapalainen, et al., 2010)

Abbildung 7 zeigt ein Beispiel für diese Art von Aufgabe. Es werden mehrere mehrstellige Zahlen gegenüber gestellt und bei Ungleichheit eine Zahlenpaares soll die Checkbox markiert werden. In der Studie wurden für jeden Schwierigkeitslevel 4 mal 5 Zahlenpaare gezeigt. (Haapalainen, et al., 2010)

Die Variation der Schwierigkeit ist hier nicht ganz so trivial wie bei den vorherigen Aufgaben. Die erste Idee ist es, die Anzahl der Ziffern pro Zahl zu erhöhen, jedoch spielt diese gar keine Rolle für den Proband, wenn dieser die Zahlen von links nach rechts vergleicht und sich die Zahlen bereits in einer der ersten Ziffern unterscheiden. Also müssen die Anzahl der Ziffern pro Zahl und gleichzeitig die Anzahl der Ziffern, die verglichen werden müssen um festzustellen, dass zwei Zahlen verschieden sind, erhöht werden. Dazu wird die Annahme für den Benutzer benötigt, dass dieser die Zahlen von links nach rechts vergleicht. Damit kann die sich unterscheidende Ziffer (im Durchschnittsfall) einfach weiter nach rechts verschoben werden, um den Schwierigkeitsgrad zu erhöhen.

3.1.5 Aufgaben zum Linien verfolgen

In dieser Aufgabenstellung geht es darum, kurvige Linien vom Start bis zum Ende zu verfolgen. Solch eine Aufgabe ist in Abbildung 8 dargestellt. Die Startpunkte sind durchnummeriert, um sie auf der rechten Seite einzutragen. Die Probanden wurden hierbei angewiesen, die Linien ausschließlich mit den Augen zu verfolgen und nicht etwa einen Finger zu Hilfe zu nehmen. Es wurde den Teilnehmern in jeder Schwierigkeitsstufe ein Schaubild mit 10 Linien gezeigt (wie in Abbildung 8 dargestellt) (Haapalainen, et al., 2010).

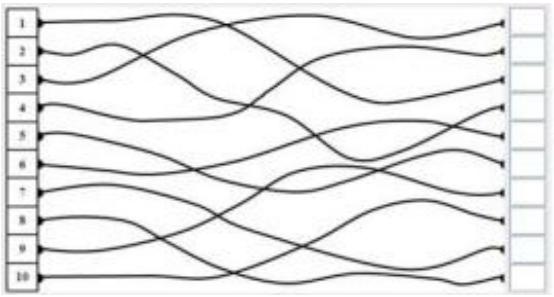

Abbildung 8: Eine mögliche Aufgabe einer Studie zur Bestimmung der Belastung mittels Linienverfolgung. Abgebildet ist eine Aufgabenstellung zur Verfolgung kurviger und überlappender Linien. (Haapalainen, et al., 2010)

Eine Veränderung der Schwierigkeit kann hier in mehreren Dimensionen geschehen (Haapalainen, et al., 2010):

- Erhöhung der Anzahl der Kreuzungen zwischen Linien.
- Erhöhung der Länge der Linien.
- Rückwärtsverfolgung wird verlangt. Dazu können beispielsweise Linien eingefügt werden, die zu keinem Endpunkt führen.

3.1.6 Aufgaben zum Buchstaben finden in einem Suchbild

In dieser Aufgabe soll ein bestimmter Buchstabe in einer Grafik verstreuter Buchstaben gesucht und markiert werden. Hierbei bieten sich auffällige Buchstaben wie beispielsweise das „x“ zum Suchen an. (Haapalainen, et al., 2010)

Eine solche Aufgabenstellung ist in Abbildung 9 dargestellt. Ziel ist es, den Buchstaben „x“ auf dem Bild zu finden und mit der Maus zu markieren. Den Probanden wurden pro Schwierigkeitsstufe 4 Suchbilder angezeigt, in denen sie den Buchstaben „x“ zu suchen hatten. (Haapalainen, et al., 2010)

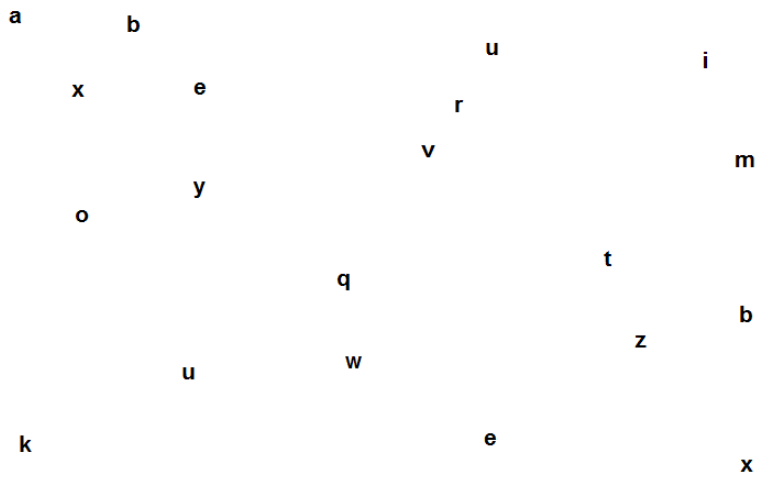

Abbildung 9: Eine mögliche Aufgabe einer Studie zur Bestimmung der Belastung mittels Buchstaben suchen, in einem Suchbild. Abgebildet ist ein Suchbild in dem Vorkommen des Buchstabens „x“ markiert werden sollen. Beispiel nach Vorlage von (Haapalainen, et al., 2010).

Die Schwierigkeitsstufe der Aufgabe lässt sich durch die Erhöhung der Anzahl der Buchstaben pro Bild erhöhen, oder indem man die Buchstaben zusätzlich sich bewegen und drehen lässt. (Haapalainen, et al., 2010)

3.2 Bestimmung der Belastung mittels physiologischer Sensoren

In (Haapalainen, et al., 2010), (Lunn & Harper, 2010), (Riener, et al., 2009), (Berguer, et al., 2001) und (Shi, et al., 2007) wurde untersucht, ob Belastung mittels verschiedener physiologischer Sensoren bestimmbar ist und mit welcher Zuverlässigkeit dies möglich ist.

Für eine genauere Differenzierung werden in (Haapalainen, et al., 2010) die Aufgaben in den Studien (siehe Abschnitt 3.1) in Gruppen eingeteilt. Die Aufgabengruppen richten sich nach der Art der logischen Schließung, die für die Lösung der Aufgabe nötig ist. Die Aufgabengruppen sind „Geschwindigkeit der logischen Schließung“, „Flexibilität der logischen Schließung“, Wahrnehmungs-Geschwindigkeit“. Zwischen den Aufgabengruppen wurden in den Studien auch tatsächlich starke Unterschiede in der Nützlichkeit der Sensoren festgestellt. In vielen Auswertungen von Zusammenhängen wurden nicht ausschließlich die reinen absoluten Sensordaten verwendet, sondern oft Ableitungen davon, wie der Median oder die Varianz.

Die Ergebnisse zur Zuverlässigkeit der Zusammenhänge fallen in den verschiedenen Studien stark unterschiedlich aus. In den umfangreichen Studien in (Haapalainen, et al., 2010) stellte sich beispielsweise heraus, dass es Zusammenhänge zwischen Sensordaten und kognitiver Last gibt. Diese Zusammenhänge sind beim Großteil der Probanden feststellbar, wenn nur genügend Sensoren verwendet werden. Jedoch sind die entscheidenden Veränderungen in den Sensorwerten bei den Probanden stark verschieden und finden auch in verschiedenen Sensoren statt. Dies macht es schwierig, einen zuverlässigen Index für kognitive Last ohne vorherige individuelle Lernphase für jeden Benutzer generieren zu können.

3.2.1 EKG / Puls

Da andere Faktoren als der Puls im EKG-Signal selten benutzt werden, wird meist deshalb auch ein bereits fertiger Pulsmesser und nicht etwa ein EKG verwendet, bei dem erst noch selbst eine Peak-Erkennung durchgeführt werden muss. Selbst bei fertigen Pulsmessern wird noch zusätzlich gefiltert und es werden beispielsweise alle Werte, die nicht im Intervall [35-155] liegen, herausgefiltert, um die Daten zuverlässiger zu machen und Ausreißer möglichst gut herauszufiltern. (Haapalainen, et al., 2010)

Mithilfe der Pulswerte konnte in der Studie von (Haapalainen, et al., 2010) lediglich eine Zuverlässigkeit von ca. 59% im Durchschnitt über 3 verschiedene Aufgabentypen erzielt werden. Bei keinem Studienteilnehmer erwies sich der Puls als sinnvollstes Merkmal zur Bestimmung der Belastung. Allerdings lässt sich durch Bestimmung der Herzfrequenzvariabilität, also die Schwankung im Momentanpuls und die anschließende Medianbestimmung über diese Differenzen bei der Aufgabe, eine Zuverlässigkeit von ca. 71% erzielen.

Die Studie von (Riener, et al., 2009) befasst sich ebenfalls mit der Herzfrequenzvariabilität Dabei geht es um die Auswertung der Herzfrequenzwerte des Fahrers eines Autos, um herauszufinden, ob diese zu einem Gradmesser für Ablenkung im Straßenverkehr taugen. Hintergrund ist, dass 25% der von der Polizei aufgenommenen Unfälle während

einer Ablenkung des Fahrers stattfinden. Ziel ist es, aus diesen Sensordaten ein Warnsignal für Unaufmerksamkeit des Fahrers zu schaffen. Wichtig für die Herzfrequenzvariabilität ist das R-R-Intervall in der EKG-Kurve und dessen Schwankungen.

Aufgrund der zwangsläufig normalen Schwankungen der geforderten Aufmerksamkeit entlang einer Strecke mussten erst Basisdaten erzeugt werden, mit denen zukünftige Werte verglichen werden können. So können in weiteren Fahrten auffällige Änderungen dieser Werte mit der zu diesem Zeitpunkt aktuellen Verkehrssituation verglichen werden. (Riener, et al., 2009)

3.2.2 Hautleitwert

Zur Verwendung des Hautleitwertes als Index für Belastung gibt es eine ganze Reihe von Arbeiten, die jedoch zu sehr unterschiedlichen Ergebnissen kommen.

Bei den schlechtesten Ergebnissen wurde lediglich eine durchschnittliche Zuverlässigkeit von 53,7% erreicht. Hierbei wurde die Varianz des Hautleitwertes verwendet, da diese noch die besten Ergebnisse lieferte. (Haapalainen, et al., 2010)

In einer weiteren Studie sollte festgestellt werden, ob eine Operation mit VES (video-endoscopic surgery) für den Chirurgen eine höhere Belastung bedeutet als ein offener Eingriff. Hierbei wurde allerdings nur die Selbsteinschätzung des Konzentrationslevels und des Stresslevels der Probanden mit dem Hautleitwert verglichen, weshalb die Ergebnisse dieser Studie nur schlecht mit den Ergebnissen anderer Studien mit objektiveren Indexen verglichen werden kann. (Berguer, et al., 2001)

In Abbildung 10 sind diese Zusammenhänge aufgezeigt: Es wird zum einen die Selbsteinschätzung bezüglich des eigenen Konzentrations- und Stresslevels und der Hautleitwert in der Pausenzeit (rest), der Zeit der offenen Chirurgie (open) und in der Zeit der VE-Chirurgie (lap), als Mittel über die Probanden gezeigt. Dabei ist sowohl eine Kurve von Probanden mit viel Operationserfahrung gezeigt und eine für Probanden mit wenig Erfahrung. (Berguer, et al., 2001)

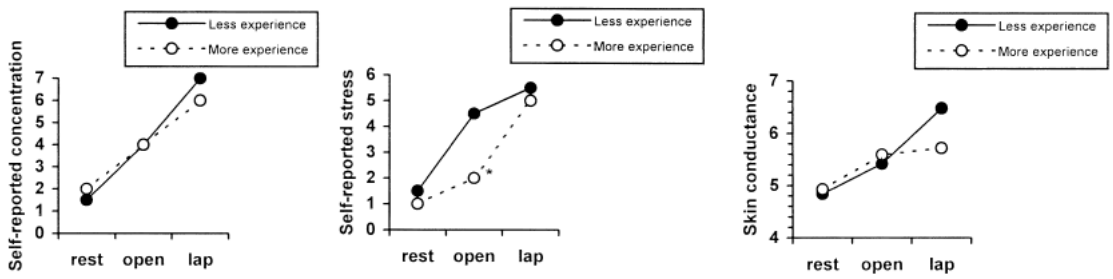

Abbildung 10: Vergleich der Selbsteinschätzung des Konzentrations- und Stresslevels und des Hautleitwertes bei einem Chirurgie-Test (Berguer, et al., 2001)

Es lässt sich durchaus eine gewisse Ähnlichkeit der Kurven erkennen, jedoch lässt sich keine Zuverlässigkeit der Werte ablesen. Aufgrund der sehr speziellen Aufgabe lässt sich dieses Ergebnis auch nicht ohne weiteres auf andere Aufgaben übertragen. Beispielsweise ist nicht gesagt, wie das Ergebnis bei körperlich weniger anstrengenden Arbeiten aussieht.

In der Studie von (Lunn & Harper, 2010) wurden Bereiche der Frustration in Web-2.0-Webseiten für ältere Benutzer festgestellt. Dies geschieht einerseits mit einem Hautleitwertsensor, um die Frustration zu erkennen und zusätzlich mithilfe eines Eye-Trackers, um den Bildschirmbereich zu erkennen.

Dabei mussten 21 Versuchsteilnehmer auf verschiedenen Seiten wie Google, einer Verbindungssuche für Züge, iGoogle und Yahoo Aufgaben durchführen. Dabei wurden Benutzer der drei Altersgruppen 30-49 Jahre, 50-59 Jahre und 60-69 Jahre untersucht. (Lunn & Harper, 2010)

Der Hautleitwert wurde an den Händen oder Füßen gemessen, da dort der Einfluss von Stress am größten ist. Trotzdem schwankten die Ergebnisse beim Hautleitwert von Benutzer zu Benutzer sehr stark. Deshalb wurden Ausschläge jeder Richtung zusammen mit der Pupillenposition festgehalten und nicht etwa nur Ausschläge nach oben. (Lunn & Harper, 2010)

Als Ergebnis wurde festgestellt, dass die kleinen Unterschiede im Hautleitwert zwischen der jüngsten Altersgruppe und der ältesten Altersgruppe sich tatsächlich vergrößern, sobald Web 2.0 Inhalte auftauchen. Daraus ergibt sich, dass die älteren Benutzer die größeren Probleme mit Web 2.0 Inhalten haben, im Vergleich zu ihren jüngeren Kollegen. Allerdings wurde auch festgestellt, dass gerade die jüngeren Versuchsteilnehmer teilweise irritiert waren, wenn keine Web 2.0 Inhalte da waren, da sie diese in viele Fällen bereits erwartet hatten. Jedoch lässt sich aus diesen durchaus interessanten Zusammenhängen nicht belegen, in wie weit der Hautleitwert als probates Mittel zur Identifizierung der momentanen Belastung eines Benutzers taugt. (Lunn & Harper, 2010)

In der Arbeit von (Shi, et al., 2007) wurde der Hautleitwert als ein Index für kognitive Last benutzt. Als Aufgabe wurde Verkehrsmanagement ausgewählt. Dabei mussten die Probanden 12 verschiedene Aufgaben pro Benutzungsschnittstelle lösen. Für die 3 verschiedene Benutzungsschnittstellen, mussten 36 verschiedene Aufgaben gestaltet werden. Ziel der Arbeit war es herauszufinden, ob der Level der kognitiven Last (abgeleitet aus dem Hautleitwert) bei multimodaler Interaktion sinkt.

Kontinuierliche Signale wie z.B. der Hautleitwertsensor eigenen sich besonders gut zur Auswertung, da hier eine prinzipiell beliebig hohe Abtastrate möglich ist und somit auch eine beliebig feine Granularität erzielt werden kann. (Shi, et al., 2007)

Als Ergebnis wurde festgestellt, dass der Hautleitwert bei multimodaler Interaktion am niedrigsten ist. Danach kommt die Interaktion per Spracheingabe. Am schlechtesten schneidet die Interaktion über Handgesten ab. Dieser Sachverhalt ist auch nochmals in Abbildung 11 in einem Diagramm in Relation gesetzt. Das gute Abschneiden der Spracheingabe im Vergleich zur Handgesteneingabe kann auch an der 100 prozentigen „Wizard of Oz“ Implementierung (und somit praktisch fehlerfreien Erkennung) der Spracheingabe liegen. (Shi, et al., 2007)

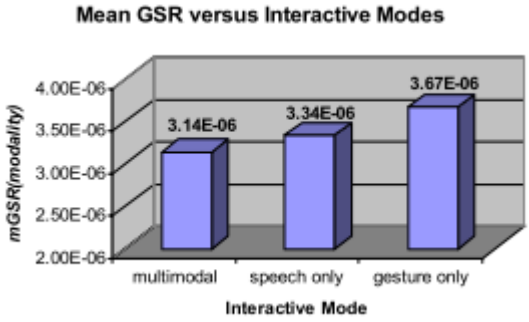

Abbildung 11: Vergleich des Hautleitwertes beim Verkehrsmanagement mittels verschiedener Schnittstellen (Shi, et al., 2007)

Es lässt sich mit dieser Studie allerdings nichts Gesichertes über die Nützlichkeit von Hautleitwertsensoren zur kognitiven Last-Bestimmung belegen, auch wenn die Annahmen bestätigt wurden und der Hautleitwert durchaus nachvollziehbare Ergebnisse geliefert hat.

3.2.3 Hauttemperatur

In der Studie von (Haapalainen, et al., 2010) konnte eine Zuverlässigkeit über die verschiedenen Aufgabentypen von 76,1% erzielt werden. Besonders bemerkenswert ist auch, dass es dabei im Vergleich zu anderen Sensoren nur geringe Schwankungen der Zuverlässigkeit zwischen den verschiedenen Aufgabentypen gab. Durch eine Kombination von Hauttemperatur und der Herzfrequenzvariabilität konnte sogar ein Index erzeugt werden, der eine Zuverlässigkeit von über 81% aufwies, mehr als durch irgendeinen einzelnen Sensor. Dabei wurden die beiden Mittelwerte über die Aufgabe addiert, um einen neuen Index zu erzeugen.

3.2.4 Gehirnströme

In der Studie von (Haapalainen, et al., 2010) wurden Gehirnströme auf ihre Nützlichkeit bezüglich der Belastungsbestimmung untersucht. Die Gehirnströme wurden mithilfe des

drahtlosen „NeuroSky mindset kit[6]“ erfasst. Dieses liefert bereits das EEG-Signal aufgeteilt in verschiedene Frequenzbereiche. Daraus wurden ebenfalls mithilfe dieses Kits zwei mentale Zustandsausgaben erzeugt. Diese waren „Aufmerksamkeit“ und „Nachdenken“.

Damit wurde allerdings nur eine Zuverlässigkeit in der Belastungsbestimmung von 60,2% erreicht. Diese recht aufwändige und sehr sensible Messung (bezüglich der Güte des Hautkontaktes der Sensoren), liegt weit hinter anderen einfacheren Messverfahren zurück. (Haapalainen, et al., 2010)

3.2.5 Muskelaktivität der Trapezmuskulatur

Bei der Studie von (Wijsman, et al., 2010) wird per Elektromyografie (EMG) die Aktivität des Trapezmuskels gemessen, um dessen Abhängigkeiten zum Stresslevel der Studienteilnehmer zu untersuchen. Zusätzlich wurden weitere Sensordaten aufgezeichnet, die in dieser Untersuchung keine weitere Beachtung fanden. Ziel war es, ein einfach zu tragendes Gerät zur Bestimmung des Stresslevels zu entwickeln. Hierbei wird zwischen zwei Arten von Stress unterschieden. Zum einen gibt es den mentalen Stresslevel, dieser wird in erster Linie durch die Probleme beim Bearbeiten der Aufgabe hervorgerufen. Zum anderen gibt es den psychosomatischen Stresslevel, der durch soziale Faktoren hervorgerufen wird. Möglichkeiten zur Erhöhung des psychosomatischen Stresslevels sind z.B. (Wijsman, et al., 2010):

- Dem Probanden wird mitgeteilt, dass nach dem ersten Fehler der Versuch für ihn vorbei ist.
- Dem Proband wird mitgeteilt, dass in Teams gearbeitet wird und dass das nächste Teammitglied dort weitermachen muss, wo er selbst aufhört bzw. nicht weiter kommt.
- Parallel laufenden Fernseher führt zu Ablenkung.
- Eine Kameraattrappe suggeriert dem Proband, dass er aufgenommen wird.
- Nach erstem Durchlauf wird dem Proband eine Belohnung versprochen, die er in den kommenden Durchläufen wieder verlieren kann.

Es wurde festgestellt, dass die Amplitude des EMG-Signals höher während Stresssituationen als während Erholungsphasen ist. Signifikante Unterschiede konnten jedoch erst durch die Berechnung des quadratischen Mittelwertes des EMG-Signals gefunden werden. Für eine genauere Analyse welche Aspekte des EMG-Signals am besten zur Stressbestimmung geeignet sind, wird allerdings auf folgende Forschungen verwiesen.

[6] http://www.neurosky.com

3.2.6 Pupillenbewegungen

In der Studie von (Haapalainen, et al., 2010) wurde untersucht, ob es Zusammenhänge zwischen Pupillenbewegungen und kognitiver Last gibt. In dieser Studie wurden sowohl die Pupillenbewegungen als auch die Änderungen in der Pupillengröße aufgezeichnet. Jeder Proband musste zuerst eine standardisierte Initialisierungsprozedur des Eye-Tracking-Systems durchlaufen.

Der Median dieser Pupillenbewegungen lieferte jedoch im Durchschnitt über verschiedene Aufgaben nur eine Zuverlässigkeit von 57,4% und damit das zweitschlechteste Ergebnis der 6 untersuchten Sensoren. (Haapalainen, et al., 2010)

3.2.7 Blinzelrate der Augen

In der Studie von (Berguer, et al., 2001) wurden Zusammenhänge zwischen der Blinzelrate der Augen und der Selbsteinschätzung des Konzentrations- und Stresslevels untersucht. Diese Werte wurden während Pausenzeiten und der Zeit eines Testes einer Operation mit offener Chirurgie und VE-Chirurgie aufgenommen.

Die Durchschnittswerte, sowohl für erfahrenere als auch für unerfahrene Chirurgen, sind in Abbildung 12 abgebildet. Dabei ist zu sehen dass die Blinzelrate sich umgekehrt proportional zur Selbsteinschätzung des Konzentrations- und Stresslevels verhält. Allerdings ist auch ein großer Ausreißer der Blinzelrate bei offener Chirurgie zu sehen. (Berguer, et al., 2001)

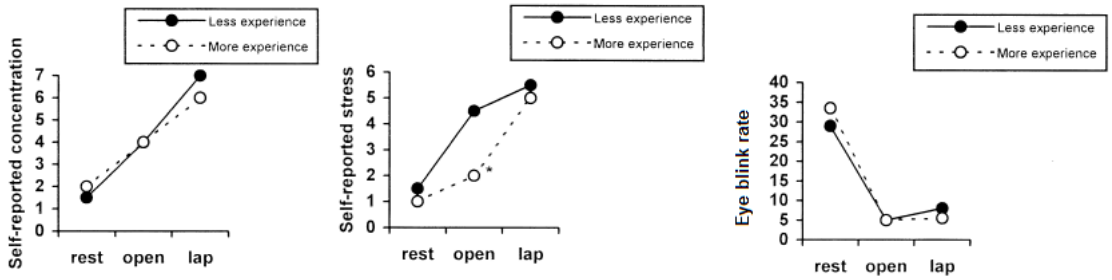

Abbildung 12: Vergleich der Selbsteinschätzung des Konzentrations- und Stresslevels und der Blinzelrate bei einer Chirurgie-Studie, die Pausenzeiten (rest), Operationen mittels offener Chirurgie (open) und Operationen mittels VES-Chirurgie vergleicht. (Berguer, et al., 2001)

3.3 Weitere Methoden zur Bestimmung der Belastung

In einigen Arbeiten wurde auch versucht, Belastung anhand anderer Methoden festzustellen. Auch diese Alternativen werden hier kurz aufgezeigt.

3.3.1 Selbsteinschätzung

In (Chen, et al., 2011) wurde die Selbsteinschätzung der kognitiven Last mit anderen Methoden der kognitiven Last-Bestimmung verglichen. In der Studie wurde neben der Selbsteinschätzung auch die Aufgabenfertigstellungsdauer (Zeit, die für die Aufgabe benötigt wurde), die Augenaktivität (Pupillenbewegung und Blinzelrate) und die Bewertung der Sorgfalt der Durchführung betrachtet.

Es wurden Aufgaben unterschiedlicher Schwierigkeit benutzt, um unterschiedliche kognitive Belastungen zu erzeugen. Die verschiedenen Methoden der kognitiven Last

-Bestimmung wurden anschließend dahingehend untersucht, ob und in wie weit sie sich mit den unterschiedlichen Schwierigkeitsstufen der Aufgabenstellungen decken. (Chen, et al., 2011)

Es wurde festgestellt, dass die Selbsteinschätzung die zuverlässigste Methode ist, obwohl sie natürlich subjektiv ist. Dabei ist zu beachten, dass die Selbsteinschätzung noch weitere Nachteile hat. Der wichtigste Nachteil ist, dass sie nicht live ausgewertet werden kann, sondern erst nach der Ausführung der Aufgabe durch den Benutzer. Der zweite Nachteil ist, dass es einen zusätzlichen Aufwand für den Benutzer bedeutet und nicht wie bei anderen Methoden einfach nebenbei, ohne die Aufmerksamkeit des Benutzers laufen kann. (Chen, et al., 2011)

Eine Möglichkeit zur Verbesserung dieser Technik wäre es zu versuchen, dem Benutzer unterbewusst durchgeführte Gesten beizubringen, dies könnte z.B. ein Kratzen am Kopf oder das Näher-an-den-Schreibtisch-rücken des Stuhles oder ähnliches sein. Dies sind Gesten, die ohnehin schon von einem Teil der Menschen unterbewusst bei kognitiver Überlast gemacht werden.

3.3.2 Aufgabenfertigstellungszeit

Im gleichen Experiment wie in Abschnitt 3.3.1 (Selbsteinschätzung) beschrieben wurde die benötigte Dauer für die Fertigstellung der Aufgabe als Index für kognitive Last benutzt.

Diese Methode lieferte zwar schlechtere Ergebnisse als die Selbsteinschätzung, ist aber leichter durchzuführen. Es handelt sich um eine sehr simple Methode die leicht zu implementieren ist und fordert kein Zutun des Benutzers. Allerdings kann sie auch nicht wirklich live Ergebnisse liefern, sondern erst nach Fertigstellung der Aufgabe, bzw. wenn der Benutzer im Vergleich zur normalen Fertigstellungsdauer bereits sehr lange Zeit gebraucht hat. (Chen, et al., 2011)

3.3.3 Analyse der Stimme

In dieser Studie wurde versucht die kognitive Last anhand von Eigenschaften der Stimme zu bestimmen. Dabei wurde die kognitive Last des Benutzers in 3 Level eingeteilt. Es wurde ein System entwickelt, das eine Vielzahl von Parametern erfordert. Deshalb musste das System anhand von Beispieldaten zuvor erst in der Analyse trainiert werden. (Yin, et al., 2007)

Die Studie wurde schließlich an 15 Personen durchgeführt, deren Muttersprache Englisch war, ansonsten wurden weitgehend zufällige Personen ausgewählt. Die Aufgabe bestand darin, erst eine Textpassage zu lesen und anschließend Fragen zu dieser Textpassage zu beantworten. In der besten Systemkonfiguration erzielte das System eine Zuverlässigkeit von 71,1% bei der Einteilung der kognitiven Last in 3 Levels. (Yin, et al., 2007)

3.3.4 Analyse der Handschrift

Diese Studie befasst sich eher mit einem Nischenthema der Informatik, der Analyse der Handschrift, hat jedoch damit auch ein Alleinstellungsmerkmal. Mit dieser Analyse wird untersucht, ob damit die kognitive Last zum Zeitpunkt des Schreibens bestimmt werden kann. Dabei wurden verschiedene Merkmale der Handschrift untersucht. Untersuchte Merkmale der Schrift waren (Yu, et al., 2011):

- Druck auf den Stift
- Geschwindigkeit des Stiftes
- Zuglängen
- Bewegungen innerhalb eines Zuges

Die Studie wurde an 20 Teilnehmern durchgeführt. Dabei wurde den Teilnehmern je nach Schwierigkeitsstufe eine Anzahl von Wörtern für eine bestimmte Dauer gezeigt. Danach mussten die Teilnehmer aus diesen Wörtern einen Satz bilden und aufschreiben. Den Teilnehmern wurde dabei nicht erlaubt, die Wörter zu notieren, bevor sie den Satz zu Papier brachten. (Yu, et al., 2011)

In der Studie wurde vor allem festgestellt, dass gerade die einfach zu analysierenden Eigenschaften wie Druck und Geschwindigkeit sehr gute Indikatoren für kognitive Last sind. Hoher Druck suggeriert eine hohe kognitive Last, wohingegen eine hohe Geschwindigkeit eine geringe kognitive Last signalisiert. Interessant ist zudem, dass die Faktoren stark voneinander abhängen, also dass hoher Druck oft auch niedrige Geschwindigkeit und niedriger Druck oft auch hohe Geschwindigkeit bedeutet. Das heißt es genügt sogar, nur einen der beiden Faktoren zu analysieren. Problematisch ist jedoch, dass diese Faktoren stark von dem geschriebenen Text abhängen und auch vor allem im Verlaufe eines Wortes stark schwanken. Diese Faktoren müssten herausgerechnet wer-

den. Ein weiteres Problem sind die Unterschiede im Schriftbild von Mensch zu Mensch. Auch schon kleine Unterschiede in der Schreibweise der Buchstaben führen zu stark unterschiedlichen Ergebnissen. Insgesamt lässt sich sagen dass die Analyse der Handschrift durchaus ein zuverlässiger Indikator für kognitive Last sein kann. (Yu, et al., 2011)

3.4 Zusammenfassung

Zusammenfassend lässt sich sagen, dass definitiv Zusammenhänge zwischen kognitiver Last und einigen physiologischen Sensoren bestehen. Insbesondere der Hautleitwert scheint dafür gut geeignet zu sein. Ob Fehlerquoten im zweistelligen Prozentbereich in einer adaptiven Benutzeroberfläche hinnehmbar sind, muss jedoch erst noch untersucht werden. In der weiteren Arbeit wird deshalb verstärkt der Fokus darauf gelegt, dass die Sensoren für eine adaptive Benutzeroberfläche verwendet werden können.

4 Konzept

In diesem Kapitel wird ein Konzept für das langfristige Ziel, eine adaptive Benutzungsschnittstelle, die sich selbständig an den Benutzer anpasst, entwickelt. Dabei soll eine Überforderung in Form von zu hoher kognitiver Last eine entsprechende Anpassung der Benutzungsschnittstelle bewirken, so dass sich die Darstellung vereinfacht und somit der Einstieg in den Lern- bzw. Verstehensprozess der Inhalte vereinfacht werden. Im umgekehrten Fall können entsprechend mehr Details zur Darstellung hinzugefügt werden.

4.1 Sensortest

Als erster Schritt wird ein Sensortest durchgeführt, um zu erfahren, was die verschiedenen Sensoren an Daten liefern. Erst danach können vernünftige Überlegungen über die weitere Verarbeitung der Daten angestellt werden.

4.2 Sensoranbindung

Im nächsten Schritt werden diese Sensoren angebunden, so dass die Sensorrohdaten zur weiteren Verarbeitung bereit stehen. Dabei muss eine geeignete Software entwickelt werden, die in der Lage ist die Sensordaten abzurufen.

4.3 Datenverarbeitung

Im nächsten Schritt müssen Teile der Rohdaten weiter verarbeitet werden. Dies kann zum einen daran liegen, dass die Rohdaten nicht direkt nutzbar sind, oder dass aus Rohdaten weitere Datenströme ableitbar sind.

Dazu gehört die Ableitung der Herzfrequenz aus dem EKG oder BVP-Signal. Dazu muss zuerst eine Peak-Erkennung implementiert werden, erst danach kann mithilfe der gefundenen Peaks die Herzfrequenz berechnet werden. Eine andere Aufgabe wäre es, gegebenenfalls Störungen aus den Signalen herauszufiltern.

4.4 Visualisierung

Auch im Hinblick auf die nachfolgende Studie müssen die (teilweise verarbeiteten) Daten visualisiert werden. Dies kann am besten in Form eines Graphs in einem Koordinatensystem geschehen. Dazu muss eine passende Bibliothek für das Live-Plotting von

Sensordaten gefunden werden. Danach muss die Live-Visualisierung für die Sensorkanäle und eventuell zusätzlich generierte Daten implementiert werden.

4.5 Studie zur Generierung von Sensordaten bei Belastung

Wenn das Programm zur Visualisierung fertig ist, kann eine erste Studie durchgeführt werden. In dieser Studie werden Zusammenhänge zwischen einigen Datenströmen und dem Belastungslevel gesucht. Nur durch solche Zusammenhänge und deren umfangreiche Auswertungen lässt sich später ein Belastungskoeffizient generieren.

In der Studie müssen sich einige Probanden an eine Auswahl der Sensoren anschließen lassen und dabei verschiedene Aufgaben durchführen. Grundsätzlich kommt dabei eine große Menge an Aufgaben in Frage. Dazu gehören unter anderem die bereits in anderen Studien angewandten und in dem Abschnitt 3.1 (Studien zur Bestimmung der Belastung) vorgestellten Aufgaben.

4.5.1 Ein Vorschlag für eine Aufgabenstellung

Damit die Studie so realistisch wie möglich die bei der Benutzung von graphischen Benutzeroberflächen auftretenden kognitiven Belastungen wiederspiegelt, werden Aufgaben an einer graphischen Benutzeroberflächen durchgeführt. Diese Aufgabengattung macht auch deshalb Sinn, da später der adaptive Prototyp auf eine verbesserte Benutzbarkeit getestet werden soll. Falls der Prototyp auf der gleichen Benutzeroberfläche aufbaut wie die in dieser Studie verwendete, können die Leistungen der Probanden dieser Studie gleichzeitig als Kontrollgruppe für die spätere Studie dienen.

Unterschiedliche kognitive Last kann durch unterschiedlich kompliziert gestaltete Aufgaben hervorgerufen werden. Zusätzlich kann die Zeit die zur Fertigstellung der Aufgabe benötigt wurde (task completion time) als Maß für die hervorgerufene kognitive Last benutzt werden. Da es bei dieser Art von Aufgabenstellung schwer ist automatisiert die kognitive Last zu ermitteln und die Selbsteinschätzung der Probanden als eine sehr zuverlässige Methode für die Ermittlung von kognitiver Last ausgemacht wurde (siehe Abschnitt 3.3.1), kann diese gleichzeitig sehr simple Methode auch in dieser Studie noch zusätzlich verwendet werden.

Die Probanden kommen nacheinander in das Versuchslabor, werden vor einen PC gesetzt und an die Sensoren angeschlossen. Zusätzlich wird der Versuch mit einer Videokamera aufgezeichnet.

Daraufhin beginnt nach einer kurzen Einweisung der Versuch. Dem Proband werden nacheinander verschiedene Aufgaben an einer einfachen graphischen Benutzeroberfläche gestellt, die er nacheinander bearbeitet. Dabei kann die kognitive Last durch divided attention Aufgaben noch zusätzlich gesteigert werden. Im Anschluss an den Test wäre

es zusätzlich noch möglich den Probanden das aufgenommene Video zu zeigen um ihren Level an kognitiver Belastung während der verschiedenen Aufgaben selbst einzuschätzen.

Für die Auswertung der Selbsteinschätzungen müssten vor allem die relativen Unterschiede in den Bewertungen der einzelnen Probanden mit den Unterschieden in den Sensorwerten verglichen werden. Ein Vergleich der absoluten Einschätzungen zwischen verschiedenen Probanden macht dagegen wahrscheinlich eher weniger Sinn. Falls solch ein Vergleich gewünscht ist, bietet es sich an die kognitive Belastung anhand der Schwierigkeit der Aufgaben, oder der task completion time zu analysieren. Die Einschätzungen werden für die spätere Auswertung in elektronischer Form festgehalten.

4.6 Generierung eines Belastungs-Koeffizienten

Aus einer Auswertung der Studie kann entschieden werden, welche Sensordaten in welcher Form in den Belastungskoeffizienten einfließen. Dabei kommen jedoch wahrscheinlich mehrere Konfigurationen in Frage, diese müssen entweder mithilfe einer weiteren Studie oder aber aus den Daten der bereits gemachten Studie auf ihre Zuverlässigkeit überprüft werden.

Der gefundene Koeffizient kann zur automatischen Bestimmung in das Programm eingearbeitet werden, so dass dieser auch mit in einem Diagramm angezeigt werden.

4.7 Prototyp einer Benutzeroberfläche, die Belastung als Eingabeparameter verwendet

Schlussendlich kann der Belastungskoeffizient noch in einem Programm verwendet werden. Dazu kann ein kleiner Prototyp, auf Basis der für die Studie verwendeten graphischen Benutzeroberfläche, erstellt werden.

Der Prototyp adaptiert sich entsprechend den Werten des Koeffizienten an die momentane Belastungssituation des Benutzers. Dazu gehört, dass die Oberfläche Hilfestellung gibt, wenn der Belastungskoeffizient einen kritischen Wert überschreitet. Dabei können entweder Hilfetexte eingeblendet werden, oder es können die Entscheidungsalternativen reduziert werden. Bei weiterhin zu hohen Belastungswerten können natürlich auch beide Vereinfachungen in Kombination angewandt werden. Optional kann auch noch in die andere Richtung entwickelt werden, das heißt, es können bei Belastungswerten unter einem Schwellenwert zusätzliche Programmfeatures angeboten werden.

4.8 Studie zur Bedienbarkeit des adaptiven Prototyps

Der erstellte Prototyp kann wiederum in einer Studie auf seine Bedienbarkeit überprüft werden.

In der Studie wird der adaptive Prototyp mit einer nicht adaptiven Variante verglichen. Dazu werden die Probanden in zwei Gruppen aufgeteilt, wobei je eine Gruppe einen der beiden Prototypen zur Bedienung bekommt. Beide Gruppen bearbeiten die gleichen Aufgaben. Danach werden die Ergebnisse der beiden Gruppen miteinander verglichen.

5 Praktische Sensoranalyse

Das Nexus10 unterstützt eine Vielzahl an physiologischen Sensoren, von denen jedoch nur eine Auswahl für einen späteren Prototypen als Eingabemodalität nutzbar ist. Dafür ist jedoch eine genauere Betrachtung der Sensoren unabdingbar. Dieser Sensortest lässt sich am einfachsten mit dem beim Nexus10 mitgelieferten Programm BioTrace+ durchführen. BioTrace+ ermöglicht es auf einfache Weise die Sensordaten anzuzeigen und teilweise diese auch schon weiter zu verarbeiten.

5.1 EKG-Sensor (ECG)

Es gibt für ein 1-Kanal EKG im Wesentlichen zwei Möglichkeiten: Eine Möglichkeit ist das Anlegen der Elektroden wie in Abbildung 13 dargestellt, wobei C die Erde ist. Die zweite Möglichkeit ist das Anlegen an den Unterarmen. Dabei besteht jedoch eine starke Sensitivität für Armbewegungen. (Mind Media B. V., 2006)

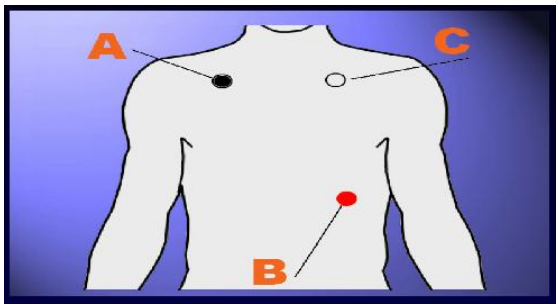

Abbildung 13: Anleitung zum Anlegen der Elektroden für ein 1-Kanal EKG mit Messung an der Brust [BioTrace+]

Ein Beispiel EKG-Signal in Ruhe ist in Abbildung 14 dargestellt. Man sieht, dass ca. jede Sekunde ein R-Peak auftaucht, was einem Puls von 60 entspricht.

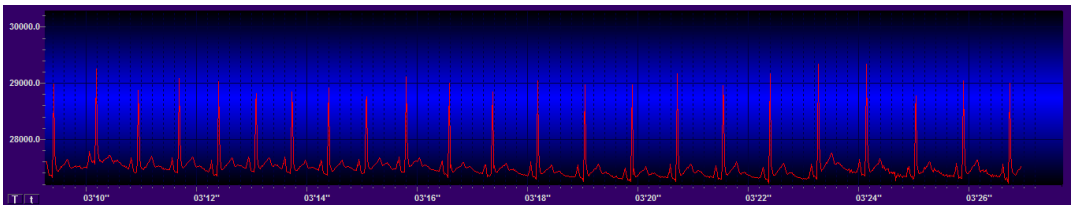

Abbildung 14: Beispiel eines EKG-Signals in BioTrace+

5.2 Hautleitwertsensor (GSR)

Der Hautleitwert macht in den ersten Funktionstests den aussagekräftigsten Eindruck. Ein Beispiel eines Signals in Ruhe ist in Abbildung 15 zu sehen.

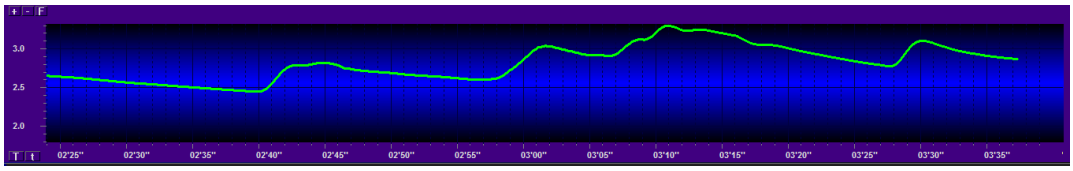

Abbildung 15: Beispiel eines Hautleitwertsignals

Die Eigenschaften dieses Sensortyps werden auch noch in einem weiteren Test bestimmt. Hierbei wird das Signal über einen längeren Zeitraum an einer zweiten Person beim Puzzeln beobachtet. Dabei konnte folgende Feststellungen gemacht werden:

- Man bemerkt einen kleinen Niveauunterschied zwischen der Aktion „ein neues Teil aus der Schachtel nehmen“ und dem Suchen einer passenden Stelle für dieses Teil, wobei der Hautleitwert beim Suchen einer passenden Stelle höher war.
- Es gibt allerdings auch viele Störfaktoren für das GSR-Signal. Dazu gehören Bewegungen als auch z.B. ein Husten, was zu starken Schwankungen führt.
- Nach einer gewissen Zeit stellt sich eine Art Resignationsphase ein, in der praktisch keine Unterschiede des Hautleitwertes zwischen verschiedenen Aktionen mehr festzustellen sind.

5.3 Blutvolumenpulssensor (BVP)

Das Signal des BVP-Sensors ist ähnlich dem des EKG-Sensors. Der BVP-Fingerclip wird an einen Finger angebracht und die Messung kann sofort starten. Damit erfordert dieser Sensor nur einen sehr geringen Aufwand zum Anlegen. Auch hier lässt sich der Puls aus dem Signal ermitteln. Die einzelnen Peaks sehen in ihrer Beschaffenheit allerdings etwas anders aus, wozu eventuell eine andere Peak-Erkennung nötig ist. Ein Beispiel eines solchen Signals ist in Abbildung 16 abgebildet. Das Signal besitzt eine gewisse Sensibilität auf Fingerbewegungen an dem Finger, an dem es angelegt ist.

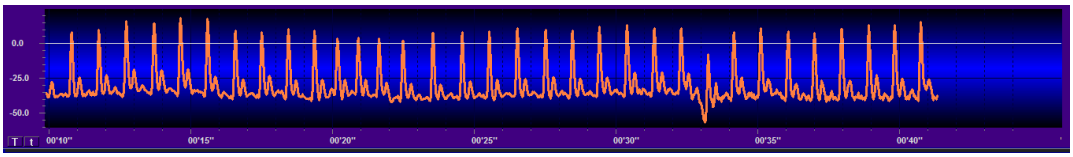

Abbildung 16: Beispiel eines BVP-Signals

5.4 Atmungs-Sensor

Der Atmungs-Sensor kann entweder für Thorax- oder Abdomen-Messung angebracht werden. Bei Abdomen-Messung muss der Sensor-Gurt auf Höhe des Bauchnabels angebracht werden.

Ein Beispiel einer Atemkurve in Ruhe einer abdominalen Messung ist in Abbildung 17 abgebildet. Aus der Atemkurve lässt sich die Atemfrequenz bestimmen, wobei dies nur

in Ruhe und ohne dass der Proband redet, hustet oder sonstiges, zuverlässig möglich scheint. Dies zeigt, dass das Signal von vielen äußeren Faktoren abhängt.

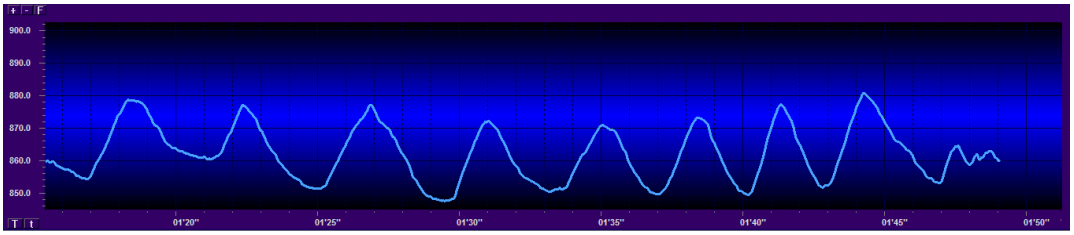

Abbildung 17: Beispiel einer Atmungs-Kurve

5.5 Gehirnstromsensor (EEG)

Der Einfachheit halber wird hier nur die 1-Kanal-Gehirnstrommessung betrachtet.

Die drei Elektroden werden auf mit Elektrodengel präparierte Hautpartien angebracht und zur sichereren Befestigung mit einem EEG-Cap gesichert.

Die Elektroden können zum Beispiel so angelegt werden:

- schwarzes Ende: hinter linkem Ohr platzieren
- rotes Ende: ganz oben auf der Mitte des Kopfes platzieren
- weißes Ende (Erde): hinter rechtem Ohr platzieren

Die Anbringung der Sensoren ist recht aufwändig. Es muss schon alleine 5-10 Minuten geartet werden, damit sich das Signal stabilisieren kann (Mind Media B. V., 2006). Aufgrund des erheblichen Zeitaufwands zur Anlage der Elektroden muss abgewogen werden, ob dieser in Relation zum prognostizierten Nutzen steht. Ein Beispiel eines solchen EEG-Signals ist in Abbildung 18 zu sehen.

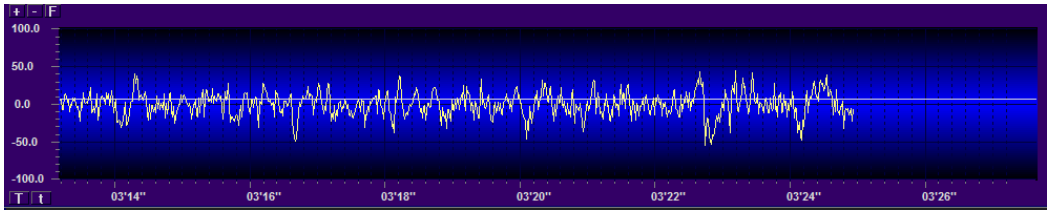

Abbildung 18: Beispiel eines EEG-Signals

Das Signal ist sehr anfällig gegenüber Bewegungen des Kopfes. Letztendlich wird über diese 1-Kanal Messung die gesamte Gehirnaktivität gemessen. Die einzige Unterscheidungsmöglichkeit sind die verschiedenen Frequenzen der Schwingungen im Signal. Da das Gehirn jedoch stark parallel arbeitet, können wir nicht feststellen, was die von der Aufgabe hervorgerufenen Aktivitäten sind und was von Hintergrundaktivitäten des Gehirns hervorgerufen wird.

6 Implementierung

In diesem Kapitel wird die Implementierung der Software für Empfang, Verarbeitung und Visualisierung der Datenströme anhand einiger wichtiger Teilaspekte des Programms erläutert.

Die Umgebung in die die Software eingebunden werden muss ist in Abbildung 19 dargestellt. Zunächst einmal werden die Sensordaten vom Nexus 10 von der CRN-Toolbox empfangen und per TCP weiterverschickt. Die Software empfängt diesen TCP-Datenstrom und wandelt die einzelnen Pakete zunächst in einzelne Datensätze in Form von programminternen Objekten um. Des Weiteren laufen parallel dazu Threads, die sich sowohl um die Peak Erkennung und Momentanpuls-Berechnung als auch um das Plotten aller Daten kümmern. Die gesamten Daten (Rohdaten und berechnete Werte) werden mithilfe des EI-Toolkits übers Netzwerk für andere Software bereitgestellt. Ein Screenshot der fertigen Software ist in Abbildung 20 dargestellt.

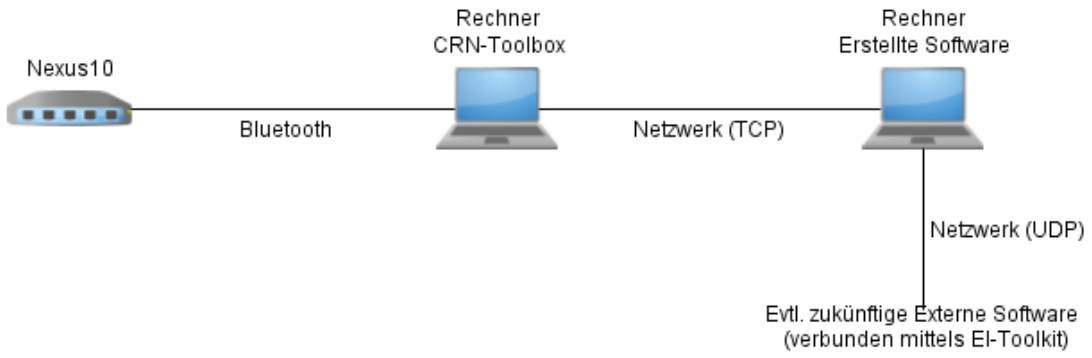

Abbildung 19: Struktur der Umgebung der erstellten Software

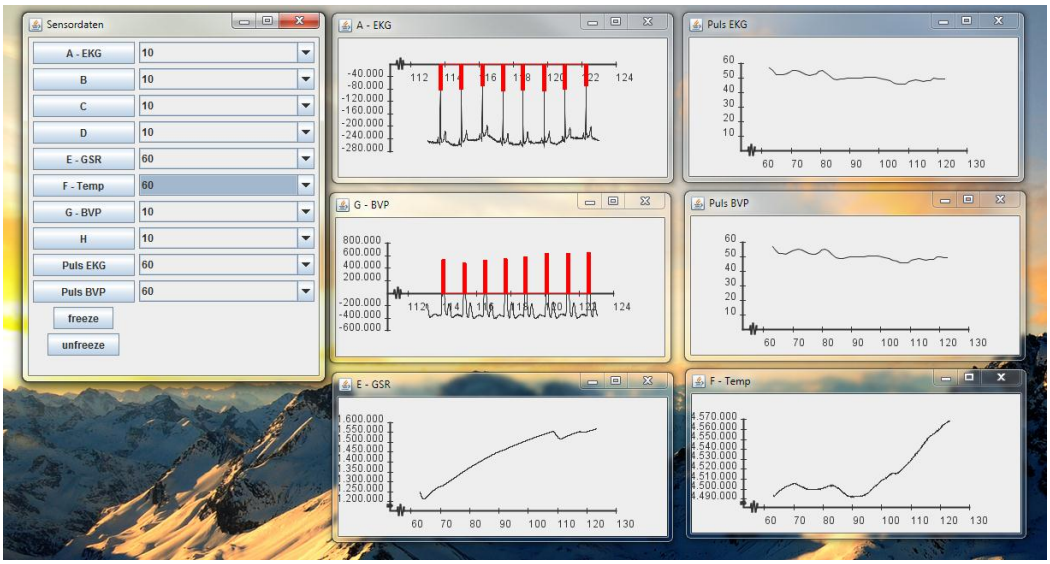

Abbildung 20: Screenshot der erstellten Software bei der Anzeige einiger Sensorkanäle und Pulssignale

6.1 Empfangen der Daten vom Nexus10 mit der CRN-Toolbox

Die CRN-Toolbox[7] (Context Recognition Network Toolbox) erlaubt es unter anderem, Daten vom Nexus10 zu empfangen. Das ist der Grund weshalb sie hier verwendet wird. Die Toolbox erlaubt es mittels unabhängiger parametrisierbarer Komponenten, Systeme für multimodale Interaktion zu erstellen.

Zuerst wurde versucht, die CRN-Toolbox unter Windows mithilfe von Cygwin[8] zum Laufen zu bekommen. Dies wurde schließlich auch erreicht. Jedoch war es nicht möglich, unter Cygwin über Bluetooth das Nexus anzubinden, so dass die CRN-Toolbox damit kommunizieren konnte. Deshalb wurde die CRN-Toolbox auf einen separaten Rechner, mit der Linux Distribution Ubuntu als Betriebssystem, installiert.

Für die Installation unter Linux sind folgende Schritte nötig:

1. SVN-Anwendung installieren (`apt-get install subversion g++ make automake1.9 autoconf libtool`)
2. Programmcode der Toolbox laden per SVN-Checkout (`svn checkout https://redmine.esl.fim.uni-passau.de/svn/crnt/trunk/toolbox`)
3. Erstellen des Konfigurationsskripts (`./autogen.sh`)
4. Makefiles generieren (`./configure --enable-wearlab`)
5. Build der Toolbox (`make`)

[7] http://crnt.sourceforge.net/CRN_Toolbox/Home.html

[8] http://www.cygwin.com/

Der nächste Schritt ist, die Konfigurationsdatei der CRN-Toolbox zu erstellen. Dazu gehört unter anderem:

- Einen Geräte-Port eintragen, mit dem nach dem Nexus10 gesucht werden soll
- Tasks erstellen (Reader vom Nexus10, TCP-Writer und evtl. File Logger und Console Writer)
- Auszulesende Kanäle des Nexus10 eintragen (In unserem Fall am besten alle; eine Auswahl kann immer noch später in der selbstentwickelten Software stattfinden.)
- IP-Adresse und Port angeben, an die der TCP-Writer senden soll

Da alles korrekt konfiguriert ist, kann das Nexus10 verbunden werden. Dazu sind folgende Schritte notwendig:

1. Nexus10 einschalten
2. Bluetooth Dongle einstecken
3. Die Mac-Adresse des Nexus10 herausfinden (z.B. mit dem Befehl `hcitool scan`)
4. Nexus10 mit einem Geräte-Port verbinden (`rfcomm bind <Port Nummer> <Mac-Adresse>`)
5. Programm (`crnt`) unter Angabe der zuvor erstellten Config-Datei starten

Nach diesen Schritten läuft das Programm und sendet einen TCP-Datenstrom an die angegebene IP-Adresse und Portnummer.

6.2 Empfangen des TCP-Datenstroms von der CRN-Toolbox

Der Empfang des von der CRN-Toolbox erzeugten TCP-Datenstroms ist der erste Schritt, der von der eigenen geschriebenen Software erledigt werden muss. Danach müssen die empfangenen Pakete noch in die einzelnen Kanäle aufgeschlüsselt werden.

6.2.1 Datenstrom empfangen

In Listing 2 ist die Methode `openConnection(...)` aus der Klasse `CRNTClient` aufgeführt. Diese Methode hat die Aufgabe, die TCP-Verbindung zur CRN-Toolbox aufzubauen.

```
1     public void openConnection(String host, int port)
2     {
3         try
4         {
5             this.clientSocket = new Socket(host, port);
6             System.out.println("clientSocket erstellt");
7             DataInputStream inStream = new DataInputStream(
8                     clientSocket.getInputStream());
9             if(inStream != null)
10            {
11                System.out.println("inStream erstellt");
12                bufferedReader = new BufferedReader(
13                        new InputStreamReader(inStream));
14                System.out.println("bufferedReader erstellt");
15                recieve = new Thread(this);
16                recieve.start();
17            }
18            else
19            {
20                System.out.println("Es konnte keine " +
21                        "Verbindung aufgebaut werden.");
22            }
23        }
24        catch(IOException e)
25        {
26            System.out.println("IO-Exception:");
27            e.printStackTrace();
28        }
29    }
```

Listing 2: Methode zum Aufbau der TCP-Verbindung zur CRN-Toolbox

Zum Verbindungsaufbau muss zunächst ein Socket erstellt werden. Dieser muss eine Verbindungsanfrage annehmen, indem ein neuer `DataInputStream` erstellt wird. Aus diesem kann ein `BufferedReader` erstellt werden, der fortan zum Empfang der Pakete dient. Die einzelnen Pakete können nach einem erfolgreichen Verbindungsaufbau mittels `bufferedReader.readLine()` gelesen werden.

6.2.2 Konvertieren der Daten in einzelne Datensätze

Ein Beispiel für einen empfangen Datenstrom ist in Listing 3 dargestellt. Dabei stellt eine Zeile einen Datensatz dar. Die empfangenen Daten hängen davon ab, was in der Konfigurationsdatei der CRN-Toolbox angegeben wurde. Die einzelnen Werte sind durch Tabs getrennt.

Es sind dabei nacheinander folgende Werte aufgeführt:

- Zeitstempel in Sekunden seit 1.1. 1970
- Zeitstempel in µs
- Signal Eingang A
- Signal Eingang B
- ...
- Signal Eingang H

- Ein Label der CRN-Toolbox (hier nicht weiter von Bedeutung)

```
1341154953 347448 -514137 0 0 0 1027131 4155595 -458750 0 0
1341154953 351441 -515022 0 0 0 1027171 4155759 -453679 0 0
1341154953 355451 -519887 0 0 0 1027209 4155656 -450590 0 0
1341154953 367446 -522983 0 0 0 1027325 4155821 -447072 0 0
1341154953 374992 -521957 0 0 0 1027372 4155438 -439305 0 0
1341154953 375015 -524097 0 0 0 1027442 4155627 -433652 0 0
1341154953 387451 -523452 0 0 0 1027531 4155518 -424855 0 0
1341154953 387484 -521550 0 0 0 1027540 4155628 -416746 0 0
1341154953 407016 -522991 0 0 0 1027576 4155445 -409452 0 0
1341154953 415450 -523299 0 0 0 1027683 4155541 -404122 0 0
1341154953 423435 -524577 0 0 0 1027679 4155459 -402129 0 0
1341154953 427435 -522762 0 0 0 1027721 4155434 -396940 0 0
1341154953 439451 -522422 0 0 0 1027860 4155416 -394074 0 0
```

Listing 3: Beispiel eines empfangenen Datenstroms, wenn alle Kanäle des Nexus10 weitergeleitet werden

Um die empfangenen Pakte weiter verarbeiten zu können, muss erst einmal das Zielformat definiert werden. Dies wurde durch die Klasse Dataset (siehe Abbildung 21) gemacht.

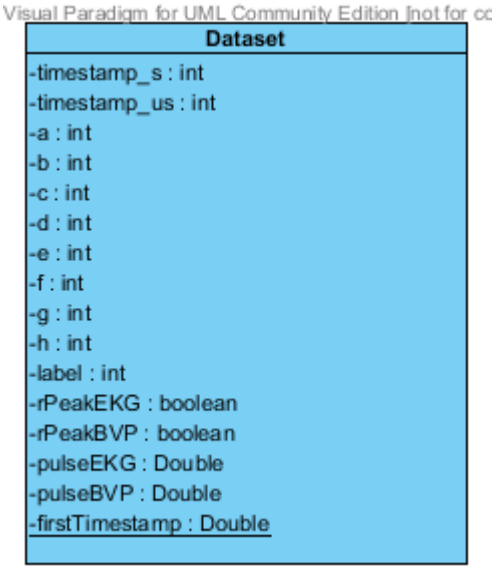

Abbildung 21: UML-Darstellung der Klasse Dataset ohne Methoden

Nun muss nur zeilenweise nach Tabs gesplittet und in Integer konvertiert werden. Dies erledigt die Methode `extractValues` der Klasse `ValueExtractor`. Der Kernabschnitt dieser Methode ist in Listing 4 dargestellt.

```
1       String[] spl = packet.split("\t");
2       int[] splInt = new int[spl.length];
3       for(int i=0; i<spl.length; i++)
4       {
5               try
6               {
7                       splInt[i] = Integer.parseInt(spl[i]);
8               }
9               catch(NumberFormatException e)
10              {
11                      //do nothing
12              }
13      }
```

Listing 4: Kernabschnitt der Methode extractValues der Klasse ValueExtractor

6.3 Visualisierung der Daten

Zur Visualisierung der Daten in Diagrammen wird die Bibliothek QN Plot[9] verwendet. Bei der Implementierung wurden erst auch einige andere Bibliotheken zum Plotten der Daten verwendet, jedoch erwies sich nur QN Plot als geeignet, um große Mengen an Daten in Realzeit zu plotten.

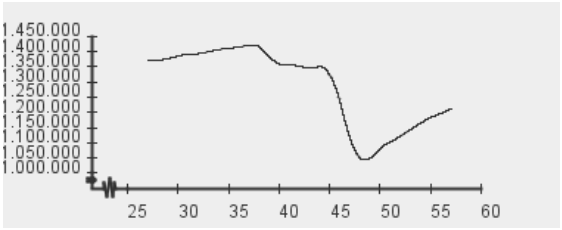

Abbildung 22: Ein Diagramm mit QN Plot erstellt

Bei der Erstellung ist zu beachten, dass dem Diagramm vom Typ `InteractiveGraph` eine `Function` zugewiesen wird und in diese die Werte eingepflegt werden. Des Weiteren müssen noch einige Initialisierungsparameter für das Diagramm gesetzt werden, dazu gehört beispielsweise das Abkürzen der Koordinatenachsen.

Eine eigene Implementierung dagegen war nötig, um die alten Werte zu entfernen und auch um die Anzeige zu aktualisieren, um regelmäßig die neuen Werte anzuzeigen. Aus Performance-Gründen kann dies nicht bei jedem neu hinzugefügten Wert geschehen. Deshalb wird dies in einem eigenen Thread ausgeführt.

[9] http://quies.net/java/math/plot/

6.4 Weiterverarbeitung der Daten

Die Weiterverarbeitung von Daten wurde in dieser Arbeit auf die Pulsberechnung anhand von EKG- und BVP-Signal beschränkt. Dazu ist als erster Schritt eine R-Peak-Erkennung nötig.

Nach Überprüfung verschiedener existenter Algorithmen zur Peak-Erkennung auf ihre Brauchbarkeit wurde entschieden, selbst einen Algorithmus zu implementieren. Keiner der bereits existenten Algorithmen erwies sich als geeignet für eine echte Live-Peak-Erkennung, da diese alle eine große Anzahl an Datensätzen für einen Durchlauf benötigen. Beispielhaft kann hier der Algorithmus von (Chernenko) dienen. Dieser arbeitet mit mehreren Filtern die nacheinander über die Daten laufen und diese glätten. Dies geschieht z.B. mit der Schnellen Fourier Transformation (FFT), angewendet als Filter um niederfrequente Komponenten herauszufiltern. Damit diese niederfrequenten Komponenten gefunden werden können muss eine ausreichende Anzahl an Datensätzen zur Verfügung stehen. Je kleiner man das Intervall wählt desto schlechter lassen sich diese niederfrequenten Artefakte erkennen, im Bereich von wenigen Sekunden sind diese praktisch nicht mehr zu erkennen.

6.4.1 Peak Erkennung

Die Peak-Erkennung wurde in der Klasse `QRS-Detector` implementiert. Es wird der Weg zum endgültigen Algorithmus anhand einiger wichtiger Entwicklungsstufen des Algorithmus zur Peak-Erkennung gezeigt. Den Anfang macht hierbei ein ganz einfacher Algorithmus, der in einigen Zyklen sowohl evolutionär als auch revolutionär weiterentwickelt wurde. In Abbildung 23 sind die von der Peak-Erkennung erkannten und anschließend visualisierten Peaks abgebildet.

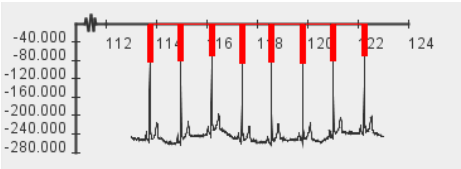

Abbildung 23: Erkannte Peaks in EKG-Daten mittels roten Balken visualisiert

6.4.1.1 Der erste Algorithmus

Im ersten Algorithmus wird eine der einfachsten Möglichkeiten für solch einen Algorithmus implementiert. Ein entsprechendes Struktogramm (Nassi-Shneiderman-Diagramm) zeigt Abbildung 24.

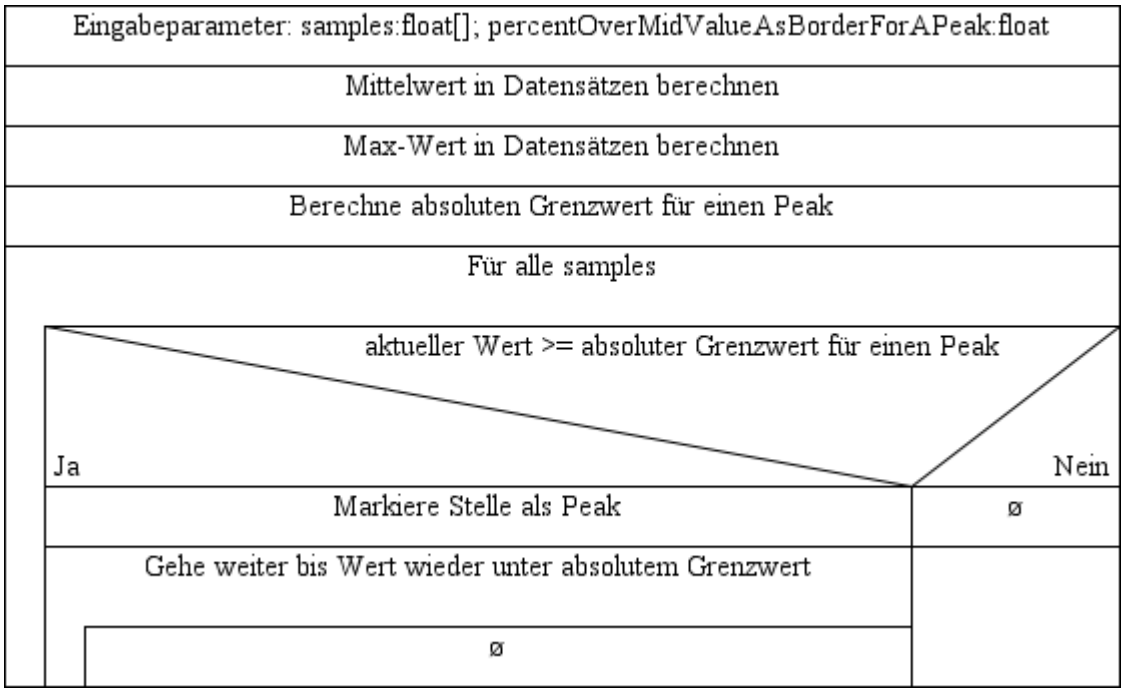

Abbildung 24: Struktogramm des ersten Algorithmus zur Peak-Erkennung

Mit diesem Algorithmus kann bereits bei guter Parameterwahl eine Trefferquote von weit über 90% erzielt werden, also dass 90% der Peaks richtig erkannt wurden. Um jedoch geringe Pulsschwankungen feststellen zu können, sollte die Trefferquote noch deutlich höher sein.

6.4.1.2 Ein zweiter Ansatz

Beim zweiten Algorithmus wird ein grundlegend anderer Ansatz gewählt, hier wird die prozentuale Steigung (im Verhältnis zwischen Mittelwert und Max-Wert) innerhalb einer festgelegten Anzahl an Datenpunkten als Schwelle für einen Peak benutzt. Dies wird in Abbildung 25 veranschaulicht.

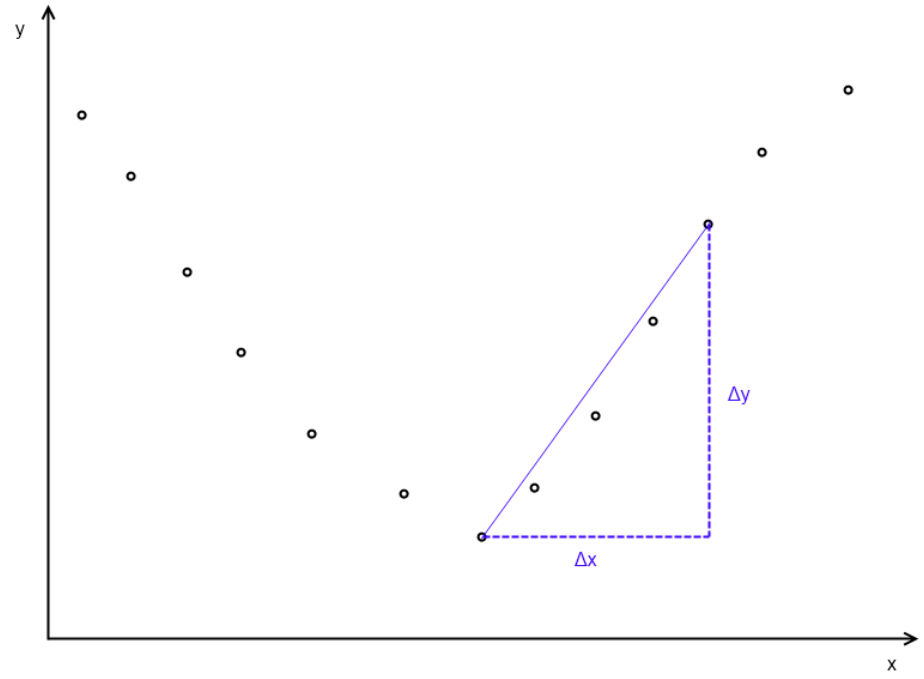

Abbildung 25: Veranschaulichung des Grenzwertes des Algorithmus zur Peak-Erkennung, in Abhängigkeit der Steigung.

Die Entscheidung, ob die gefundene Steigung Δy zu einem Peak ausgewertet wird, wird mit folgender Bedingung entschieden:

$$\Delta y \geq d * (y_{max} - y_{\emptyset})$$

Formel 2: Bedingung für einen Peak im Peak-Erkennungs-Algorithmus.

In Abbildung 26 ist der Algorithmus wiederum durch ein Struktogramm schematisch dargestellt.

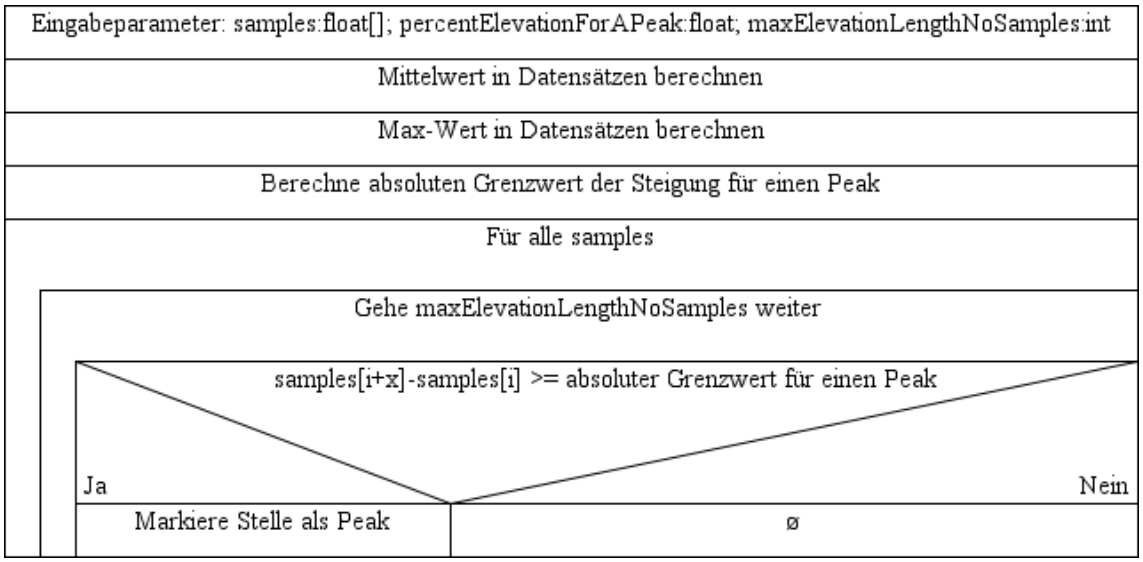

Abbildung 26: Struktogramm des zweiten Algorithmus zur Peak-Erkennung

Dieser Algorithmus erzielt deutlich bessere Trefferquoten als der erste Algorithmus. Auch werden ordentliche Trefferquoten in einem deutlich größeren Bereich in der Parameterwahl erzielt.

6.4.1.3 Optimierung des zweiten Ansatzes

Da der zweite Ansatz im Vergleich zum ersten erfolgversprechender war, wird dazu übergegangen diesen Schritt für Schritt zu verfeinern.

Die Verfeinerung liegt darin, dass, nachdem ein Peak gefunden wurde, weiter bis zum Ende des Peaks gesprungen wird. Das Ende des Peaks wird hierbei durch den ersten folgenden Wert definiert, der unter dem vorherigen liegt. Damit wird neben dem Gewinn an Performanz dafür gesorgt, dass an einem Peak nicht zwei aufeinanderfolgende Peaks erkannt werden.

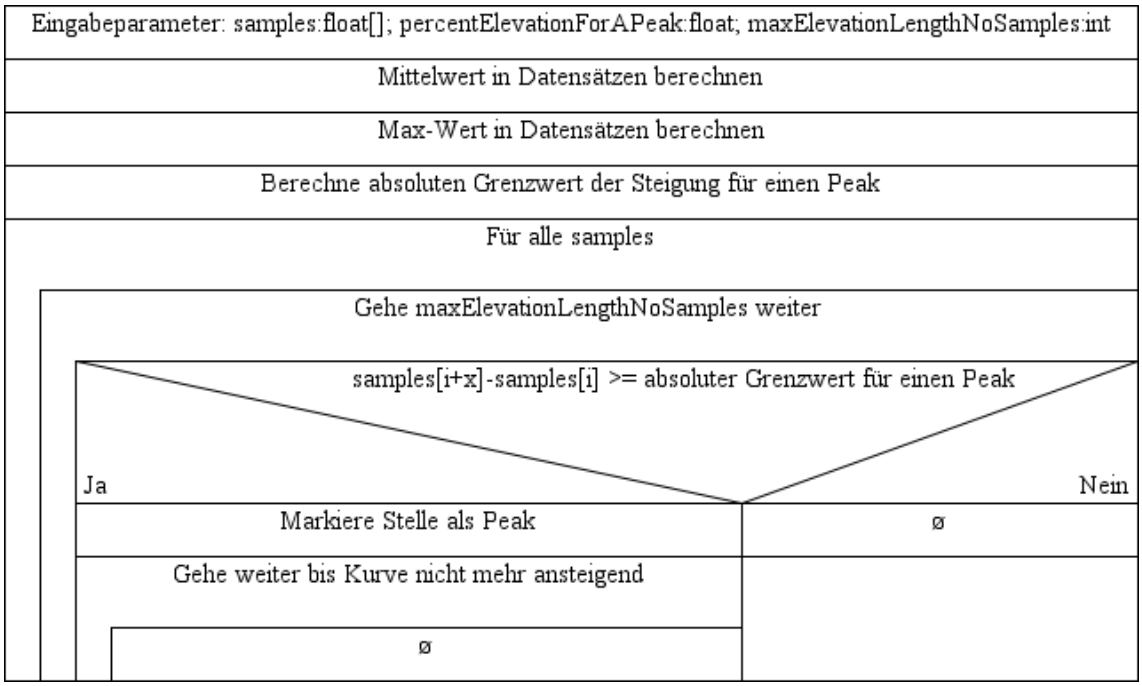

Abbildung 27: Struktogramm des optimierten Algorithmus zur Peak-Erkennung

6.4.1.4 Erhöhung der maximalen Aufruf-Frequenz des Algorithmus

Das Problem der bisherigen Algorithmen ist, dass die Aufruf-Frequenz dadurch begrenzt wurde, dass mindestens ein Peak in den übergebenen Daten enthalten sein muss, da sonst der Max-Wert nicht auf einem Peak liegen würde. Damit würden der Mittel- und Maximalwert zu nahe beieinander liegen und es würden falsche Peaks gefunden.

Die einfachste Lösung war zunächst bei jedem Aufruf zusätzlich eine ausreichende Zahl alter Daten als Eingabeparameter zu übergeben. Die Ausgabe musste noch auf doppelte bzw. nahe beieinander liegende Peaks gefiltert werden. Allerdings ergaben sich bei dieser Lösung, gerade bei mehr als einer laufenden Peak-Erkennung, Performance-Probleme.

Eine weitere Lösung benötigt als Eingabeparameter einen absoluten Wert für die Schwellen-Steigung für einen Peak. Dieser Parameter hätte jedoch zu Beginn jeder Messung neu angepasst werden.

6.4.1.5 Finaler Algorithmus

Für den finalen Algorithmus wurde noch einmal die gesamte Programmstruktur neu überdacht um die Vorteile von 6.4.1.3 und 6.4.1.4 zu kombinieren. Der vollständige Algorithmus wird im Folgenden in seinen Eckpunkten beschrieben, insbesondere die Verwaltung der Zwischenspeicherungsstrukturen bleibt hier der Einfachheit halber unberücksichtigt.

Die vorherigen Algorithmen waren in static-Methoden implementiert. Dieses Konzept wurde aus Gründen der Flexibilität und Performanz verworfen. Für jede Peak-Erkennung muss ein eigenes Objekt der Klasse `QRSDetector` instanziiert werden. In dieser muss die Methode `startQRDetection` mit den entsprechenden Parametern aufgerufen werden. Die Methode `startQRDetection` greift selbstständig auf die `Database` zu und holt sich die neuesten Datensätze von dort ab. Der Kern des Algorithmus ist in Listing 5 vereinfacht in Form von Pseudocode dargestellt.

```
1     IF (Ringbuffer voll
2                 OR Genug Werte fuer ersten Durchlauf vorhanden)
3           Rufe neue Daten aus Database ab;
4           IF (Ringbuffer voll)
5                 Verwende Algorithmus aus 6.4.1.3;
6           ELSE
7                 Verwende Algorithmus aus 6.4.1.4;
8           Ueberpruefe gefundene Peaks auf
9                 realistische RR-Intervalle;
10          Schreibe gefundene Peaks in Ringbuffer;
11          Berechne Puls;
```

Listing 5: Finaler Algorithmus zur Peak-Erkennung in Pseudocode. Implementiert in der Methode startQRDetection der Klasse QRSDetector.

Beim Start des Algorithmus wird der Algorithmus wie in Abschnitt 6.4.1.3 beschrieben verwendet, allerdings mit der Änderung, dass im Rückgabewert auch die festgestellten Steigungen der Peaks zurückgegeben werden. Nach diesem Aufruf werden die Steigungen und die zugehörigen Zeitmarken der Peaks in einen Ringpuffer geschrieben. Nach Ausführung des Algorithmus wird überprüft, ob Peaks zu nahe beieinander liegen, gegebenenfalls werden diese verworfen. Dies wird solange fortgeführt bis der Ringpuffer, dessen Größe in einer Konstante der Klasse `Constants` gespeichert ist, die maximale Größe erreicht hat.

Sind die Ringpuffer gefüllt, wird aus den gespeicherten Peaks die mittlere Steigung berechnet. Anhand dieses Wertes und der dem Algorithmus als Parameter übergebenen relativen Schwellen-Steigung für einen Peak, kann eine absolute Schwellen-Steigung

berechnet werden. Mit diesem Wert ist es möglich den Algorithmus mit absolutem Grenzwert wie in Abschnitt 6.4.1.4 ohne die beschriebenen Nachteile aufzurufen. Auch an diesem Algorithmus musste noch die Änderung vorgenommen werden, dass die festgestellten Steigungen der Peaks zurückgegeben werden, denn sonst könnte die mittlere Steigung nicht mehr ständig aktualisiert werden. Der Algorithmus kann von nun an bei ausreichender Rechenleistung theoretisch mit jedem neuen Datensatz erneut angestoßen werden und ist somit, bis auf die Berechnungszeit, verzögerungsfrei. Nach jeder Peak-Erkennung, bei der ein Peak gefunden wurde, wird eine Pulsberechnung wie in Abschnitt 6.4.2 beschrieben angestoßen.

6.4.2 Momentanpulsberechnung

Da die Peak-Erkennung bereits die Peaks filtert, kann für die Pulsberechnung ein simpler Algorithmus angewandt werden. Der vollständige Algorithmus ist in Listing 6 dargestellt. Dem Algorithmus werden zusammenhängende Zeitstempel übergeben. Da die Zeitmarken der Peaks bereits auf einen Mindestabstand überprüft wurden, muss dies hier nicht mehr geschehen. Werden nicht mindestens 2 Zeitstempel übergeben kann keine Pulsberechnung stattfinden. In diesem Fall wird als Puls Wert `NaN` (Not a Number) zurückgegeben. Andernfalls wird die Anzahl der Werte abzüglich des ersten Peaks durch den Zeitraum in Minuten zwischen erstem und letztem Peak geteilt. Das Ergebnis ist der Momentanpuls bezüglich der übergebenen Datensätze. Andernfalls wird der Puls nach folgender Formel berechnet:

$$Puls = \frac{Anzahl\ \text{übergebener}\ Peaks}{\frac{Letzter\ Peak\ (in\ s) - Erster\ Peak\ (in\ s)}{60}}$$

Formel 3: Pulsberechnungsformel im verwendeten Algorithmus

Diese Formel bildet das Kernstück des Algorithmus zur Pulsberechnung in der Methode `calculatePulse`:

```
1     public static double calculatePulse(Double[] timeVals)
2     {
3         double pulse = Double.NaN;
4         if(timeVals.length > 1)
5             pulse = (timeVals.length-1)
6                 /((timeVals[timeVals.length-1]-timeVals[0])
7                         /60d);
8
9         return pulse;
10    }
```

Listing 6: Methode calculatePulse der Klasse QRSDetector

6.5 Konfigurationsdatei auslesen

Da einige Parameter des Programms per Konfigurationsdatei konfigurierbar sein sollen, mussten dafür geeignete Klassen erstellt werden. Als Format für die Konfigurationsdatei wurde das INI-Datei-Format gewählt.

Zur Implementierung der Klasse `INIFileReader` wurde die Bibliothek ini4j[10] verwendet. Diese ermöglicht auf einfache Art und Weise eine Datei im INI-Datei-Format auszulesen. Die INI-Datei wird dabei, wie in Listing 7 zu sehen, geladen.

```
1    Ini ini = new Ini();
2    Config conf = new Config();
3    ini.setConfig(conf);
4    ini.setFile(new File("config.ini"));
5    ini.load();
```

Listing 7: Laden einer INI-Datei mit ini4j

Die einzelnen Optionswerte folgendermaßen mithilfe des erstellten INI-Objektes wie folgt abgerufen werden:

```
1    String value = ini.get("Sektionsname", "Optionsname");
```

Listing 8: Optionswert aus einem Ini-Objekt laden

Eine einfache INI-Datei ist in Listing 9 aufgeführt. Darin ist zu sehen wie Sektionen und Optionen in der INI-Datei definiert werden.

```
1    [Sektion1]
2    Option1 = Wert1
3    Option2 = Wert2
4    [Sektion2]
5    Option1 = Wert3
6    Option2a = Wert4
```

Listing 9: Aufbau einer einfachen INI-Datei

6.6 Anbindung an EI-Toolkit

Um die von dieser Software abgerufenen Sensordaten und weiterverarbeiteten Werte, auf schnelle und einfache Weise verfügbar zu machen, werden die Daten mit dem EI-Toolkit[11] ins Netzwerk geschickt. Das EI-Toolkit sendet die Daten per Broadcast ins Netzwerk und kann an anderer Stelle, ebenfalls mit dem EI-Toolkit wieder abgerufen werden. Diese Funktionalität wurde in der Klasse `EIToolkitConnector` implementiert.

[10] http://ini4j.sourceforge.net/

[11] http://svn.hcilab.org/EIToolkit/EIToolkit2/

Um das EI-Toolkit in Java verwenden zu können, muss die Bibliothek eingebunden und die beiden DLL-Dateien `eitoolkit_java.dll` und `EIToolit.dll` ins Projektverzeichnis gepackt werden. Für die Verbindung ins Netzwerk muss erst ein `Sender` initialisiert werden. Dazu sollte auch eine `Description` erstellt werden, damit auch versendet wird, welche Daten gesendet werden. Danach wird ein eigener Thread, der zyklisch die Methode `sendDataset(Dataset ds)` aufruft, gestartet. Die Methode `sendDataset` versendet einen einzelnen `Dataset` mit dem EI-Toolkit. Die wesentlichen Ausschnitte von `sendDataset` sind in Listing 10 aufgeführt.

```
1       private void sendDataset(Dataset ds)
2       {
3           DataMessage msg = sender.createDataMessage();
4
5           msg.setDouble(ConfigFileContents.getInstance().
6                   getLabelPlotterA(), ds.getA());

...

27          sender.sendMessage(msg);
28      }
```

Listing 10: Methode sendDataset der Klasse EIToolkitConnector die einzelne Datensätze mit dem EI-Toolkit versendet

Wie ein Empfänger für die versandten Daten aussehen kann ist in der Klasse `EIToolkitReceiver` implementiert. In der zugehörigen Main-Methode ist ein eigenständiges Programm implementiert, das testweise die empfangenen Daten in der Konsole ausgibt. Diese Main-Methode ist in Listing 11 dargestellt.

```
1     public static void main(String[] args)
2     {
3         DataListener listener = new DataListener()
4         {
5             @Override
6             public void onMessage(DataMessage msg)
7             {
8                 String sender = msg.getSender();
9                 double a = msg.getDouble("A - EKG");
10                double b = msg.getDouble("B");
11                double c = msg.getDouble("C");
12                double d = msg.getDouble("D");
13                double e = msg.getDouble("E - GSR");
14                double f = msg.getDouble("F - Temp");
15                double g = msg.getDouble("G - BVP");
16                double h = msg.getDouble("H - RSP");
17                double pulseEKG = msg.getDouble("Pulse EKG");
18                double pulseBVP = msg.getDouble("Pulse BVP");
19
20                System.out.println(sender + ": " + a
21                        + " " + b
22                        + " " + c
23                        + " " + d
24                        + " " + e
25                        + " " + f
26                        + " " + g
27                        + " " + h
28                        + " " + pulseEKG
29                        + " " + pulseBVP);
30            }
31        };
32        Receiver receiver = new Receiver(new StringMap());
33        receiver.addDataListener(listener);
34
35        Scanner sc = new Scanner(System.in);
36        sc.nextLine();
37        receiver.delete();
38    }
```

Listing 11: Main-Methode der Klasse EIToolkitReceiver die die versandten Daten des EIToolkitConnector empfängt und ausgibt.

7 Evaluierung

Nachdem die Software vollständig erstellt ist, wird in diesem Kapitel die Funktionalität der erstellten Software dargelegt.

7.1 Funktionalität der erstellten Software

Während der Schwerpunkt in Kapitel 6 mehr auf den technischen Details, die im Hintergrund laufen, lag, wird der Fokus in diesem Kapitel auf der Software als Ganzes liegen. Dabei wird die Funktionalität der Software als geschlossenes System analysiert.

7.1.1 Allgemeine Funktionalität

Das Programm öffnet nach dem Starten ein Fenster und versucht selbständig eine TCP-Verbindung zur bereits laufenden CRN-Toolbox aufzubauen. Nachdem die Verbindung steht kann für jeden Eingang ein Fenster für das jeweilige Diagramm der Sensordaten geöffnet werden. Weiterhin gibt es die Möglichkeiten alle Diagramme in ihrem aktuellen Zustand „einzufrieren“ (freeze) und dies auch wieder aufzuheben (unfreeze). Auch nach dem „einfrieren“ werden weiterhin Daten empfangen und die Peak-Erkennung läuft im Hintergrund weiter.

Interessant sind auch die Funktionen die direkt in den Diagrammfenstern angeboten werden. Dort kann das Diagramm in verschiedenen Formaten exportiert werden. Auch ein Drucken des Diagramms ist direkt möglich.

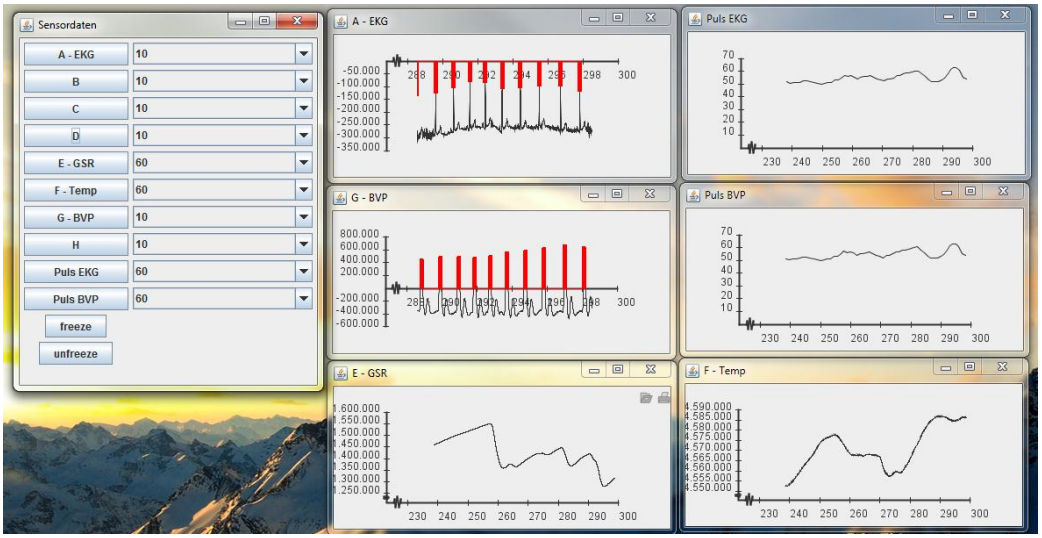

Abbildung 28: Screenshot der erstellten Software im Betrieb

7.1.2 Peak Erkennung

An der Peak-Erkennung, wie in Abschnitt 6.4.1 dargestellt, wurde viel optimiert. Die Ergebnisse werden in diesem Abschnitt dargestellt.

Die Peaks werden durch die in Abbildung 29 und Abbildung 30 dargestellten roten Balken markiert. Diese wurden ursprünglich nur zu Testzwecken eingeblendet, schlussendlich jedoch auch in der endgültigen Software belassen.

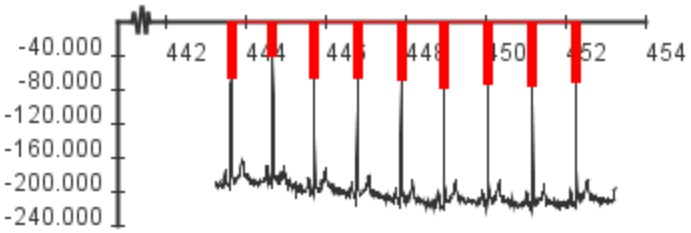

Abbildung 29: Ausschnitt aus einem EKG-Signal mit erkannten Peaks

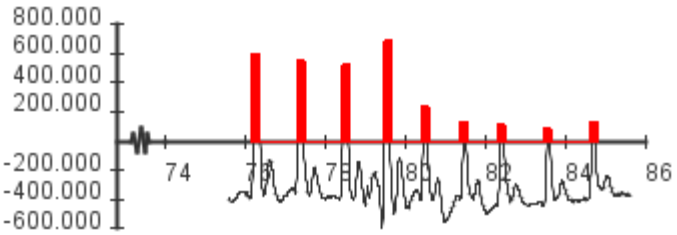

Abbildung 30: Ausschnitt aus einem BVP-Signal mit erkannten Peaks

In einem kurzen allerdings nicht repräsentativen Test wurde die Zuverlässigkeit der Peak-Erkennung überprüft. Dabei wurde die Anzahl der erkannten Peaks automatisiert ermittelt und die Anzahl der falsch erkannten und nicht erkannten Peaks von Hand gezählt. Die Ergebnisse sind in Tabelle 1 dargestellt.

	EKG	BVP
Peaks insgesamt:	425	425
Erfasste Peaks:	421 (≈99%)	413 (≈97%)
Davon falsch erfasste Peaks:	0 (=0%)	1 (≈0,2%)
Nicht erfasste Peaks:	4 (≈1%)	13 (≈4%)

Tabelle 1: Ergebnis der Zuverlässigkeitsmessung der Peak-Erkennung

7.1.3 Pulserkennung

Da die Erkennung des Pulses ausschließlich von der Genauigkeit der Peak-Erkennung abhängt, wird hier nur ein kleiner Test durchgeführt. Das Problem hierbei ist, dass ein zweites System zur Pulserkennung benutzt werden muss. Hiervon sind aber nicht alle Parameter vorhanden. Solch ein Parameter wäre beispielsweise, wie schnell eine Ände-

rung in der Herzfrequenz angezeigt wird. Ein anderer Parameter ist wie viele der letzten inter-beat Intervalle für die Pulsberechnung verwendet werden und evtl. auch welche Kriterien vorhanden sind, damit einige Intervalle nicht berücksichtigt werden. In der hier erstellten Software werden beispielsweise die letzten 4 inter-beat Intervalle berücksichtigt und Peaks, die weniger als 0,25 Sekunden nach dem vorherigen liegen (entspricht einer Herzfrequenz > 240 Schlägen/Minute), herausgefiltert. Aus diesem Grund wird jeder Vergleich von Pulswerten unterschiedlicher Systeme keine vergleichbaren Ergebnisse liefern, sobald leichte Schwankungen im Puls auftreten.

Um eine größtmögliche Vergleichbarkeit zu gewährleisten wird der Ruhepuls zur Analyse verwendet. Dieser wird von den zu vergleichenden Systemen parallel gemessen und in definierten Zeitintervallen von einer zweiten Person notiert. Die Werte werden immer im Abstand von 10 Sekunden aufgezeichnet. Die Pulswerte der selbst entwickelten Pulserkennung, die auch Kommastellen enthalten, werden kaufmännisch gerundet. Als Vergleichssystem wird ein Wahoo[12] Pulsgurt mit Ant+ in Verbindung mit dem Garmin[13] Forerunner 405 verwendet. Die Pulswerte dieses Experiments sind in Abbildung 31 in Diagrammform dargestellt. Beim Diagramm ist der Wertebereich der X-Achse auf das wesentliche Intervall beschnitten.

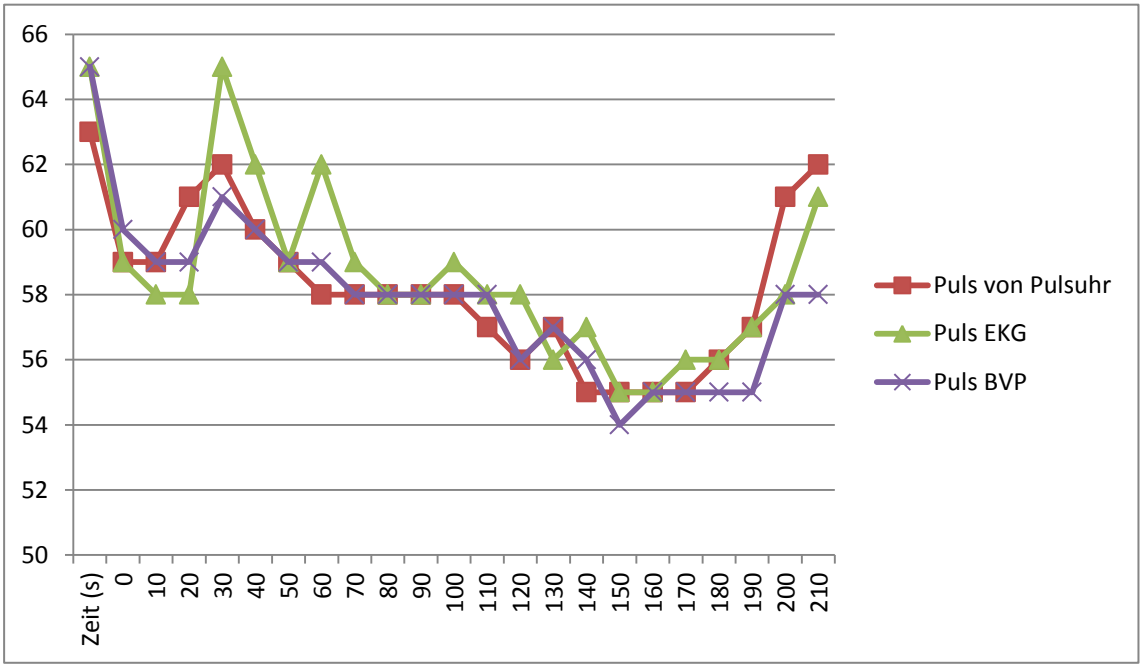

Abbildung 31: Diagramm von Pulswerten dreier verschiedener Verfahren

Anhand der Mittelwerte lässt sich die Abweichung am besten in Zahlen darstellen. Beim EKG-Puls gibt es eine Abweichung von 0,43 Schlägen/Minute und beim BVP-Puls von 0,35 Schlägen/Minute. Dies zeigt, dass die oben beschrieben Probleme der Vergleichbarkeit stark reduziert werden konnten, da das Experiment in Ruhe und damit

[12] http://www.wahoofitness.com/

[13] http://www.garmin.com

bei einem relativ stabilen Puls durchgeführt wurde. Zu beobachten ist allerdings, dass die Werte der Pulsuhr weniger stark schwanken, als die selbst errechneten. Dies könnte allerdings leicht durch eine veränderte Parameterwahl, für die Anzahl der berücksichtigten inter-beat Intervalle, geändert werden. Ein Nachteil wäre allerdings, dass der angezeigte Puls dann mehr von der Vergangenheit abhängt.

8 Zusammenfassung und Ausblick

Ziel dieser Arbeit war das Untersuchen der automatischen Adaption von Benutzeroberflächen anhand der kognitiven Belastung des Benutzers. Der Schwerpunkt lag dabei auf der Adaption anhand physiologischer Sensoren.

Dieses sehr breite und vielschichtige Themengebiet wurde, nach einigen Hintergrundthemen, in Kapitel 3 (Verwandte Arbeiten) betrachtet. Die verwandte Literatur wurde dabei auf ihre Aussagen bezüglich Studienaufbau und -ablauf sowie deren Ergebnisse untersucht. Die physiologischen Sensoren wurden gemäß ihrer Brauchbarkeit für die Arbeitslastbestimmung untersucht.

Des Weiteren wurde ein erster Schritt in Richtung einer adaptiven Benutzungsschnittstelle mit physiologischen Sensoren als Eingabemodalität gemacht. Wobei die Anbindung, Verarbeitung und Visualisierung der Daten des Nexus10-Biofeedbackgerätes die wesentlichen Punkte darstellte. Die Sensordaten werden zunächst von der entsprechend konfigurierten CRN-Toolbox per Bluetooth empfangen. Von dort werden die Sensordaten in einem TCP-Datenstrom weitergeleitet. Diesen Datenstrom greift die hier vorgestellte Software ab. Diese Daten werden gespeichert und können, im Fall von EKG- und BVP-Daten, einer Echtzeit-Peak-Erkennung zur Verfügung gestellt werden. Die Peak-Erkennung hat die Eigenschaft, bis auf die benötigte Rechenzeit, verzögerungsfrei zu sein. Aus den berechneten Peaks wird anschließend ein Live-Puls berechnet. Die Rohdaten werden mitsamt den Ergebnissen der Berechnungen in einem Live-Graph visualisiert, der auch die Möglichkeit des Exports desselben bietet. Die Einstellungsmöglichkeiten bezüglich der Live-Pulsberechnung werden zur einfachen und schnellen Konfigurierbarkeit aus einer Konfigurationsdatei ausgelesen. Um die berechneten Live-Ergebnisse leicht in anderen Projekten verwenden zu können, werden die Daten mithilfe des EI-Toolkits übers Netzwerk bereitgestellt.

Die Basis für die Entwicklung von adaptiven Benutzeroberflächen mit kognitiver Belastung als Eingabemodalität wurde mit dieser Arbeit gesetzt. Für tiefgreifendere Ergebnisse werden weitere Arbeiten benötigt, für die die Ergebnisse dieser Arbeit als Basis dienen können.

Weitere Arbeiten sollten damit beginnen, in einer Studie Zusammenhänge zwischen kognitiver Last und Stress herzustellen. Ein Vorschlag für eine solche Studie wurde bereits in Abschnitt 4.5.1 gemacht. Eine Analyse der Ergebnisse dieser Studie kann es ermöglichen einen zuverlässigen Stresskoeffizienten zu bestimmen, um einen Prototy-

pen für eine adaptive Benutzeroberfläche zu erstellen. Als Basis kann das Konzept eines solchen Prototyps aus Abschnitt 4.7 dienen.

Literaturverzeichnis

Berguer, R., Smith, W. & Chung, Y., 2001. Performing laparoscopic surgery is significantly more stressful for the surgeon than open surgery. *Surgical Endoscopy,* Band 15, pp. 1204-1207.
http://dx.doi.org/10.1007/s004640080030

Chen, S., Epps, J. & Chen, F., 2011. *A comparison of four methods for cognitive load measurement.* New York, NY, USA, ACM, pp. 76-79.
http://dx.doi.org/10.1145/2071536.2071547

Chernenko, S., kein Datum *ECG processing - R-peaks detection - Librow - Software.* [Online]
Available at: http://www.librow.com/cases/case-2
[Zugriff am 9 November 2012].

Fleschner, F., 2012. Roboter fühlen Stress. *Focus*, Juli, p. 78.

Haapalainen, E., Kim, S., Forlizzi, J. F. & Dey, A. K., 2010. *Psycho-physiological measures for assessing cognitive load.* New York, NY, USA, ACM, pp. 301-310.
http://dx.doi.org/10.1145/1864349.1864395

Lunn, D. & Harper, S., 2010. *Using Galvanic Skin Response Measures To Identify Areas of Frustration for Older Web 2.0 Users.* New York, NY, USA, ACM, pp. 34:1--34:10.
http://dx.doi.org/10.1145/1805986.1806032

Mind Media B. V., 2006. *Nexus-10 Handbuch.* s.l.:s.n.

National Heart Lung and Blood Institute, 2010. *What Is an Electrocardiogram?.* [Online]
Available at: http://www.nhlbi.nih.gov/health/health-topics/topics/ekg/
[Zugriff am 01 November 2012].

Novick, D. G. & Ward, K., 2006. *Why don't people read the manual?.* New York, NY, USA, ACM, pp. 11-18.
http://dx.doi.org/10.1145/1166324.1166329

Oviatt, S., 2006. *Human-centered design meets cognitive load theory: designing interfaces that help people think.* New York, NY, USA, ACM, pp. 871-880.
http://dx.doi.org/10.1145/1180639.1180831

Prof. Dr. Volker Claus, P. D. A. S., 2006. *Duden Informatik A-Z Fachlexikon für Studium, Ausbildung und Beruf.* Mannheim: Dudenverlag (Brockhaus AG).

Riener, A., Ferscha, A. & Aly, M., 2009. *Heart on the road: HRV analysis for monitoring a driver's affective state.* New York, NY, USA, ACM, pp. 99-106. http://dx.doi.org/10.1145/1620509.1620529

Shi, Y. et al., 2007. *Galvanic skin response (GSR) as an index of cognitive load.* New York, NY, USA, ACM, pp. 2651-2656. http://dx.doi.org/10.1145/1240866.1241057

Sung, M. & Pentland, A., 2005. Minimally Invasive Physiological Sensing for Human-Aware Interfaces. *HCI International*.

Surmann, P. D. H. & Worst, R., 2012. *Frauenhofer Institut.* [Online] Available at: http://www.iais.fraunhofer.de/nifti.html [Zugriff am 5 10 2012].

Sweller, J., van Merrienboer, J. J. G. & Paas, F. G. W. C., 1998. Cognitive Architecture and Instrumental Design. *Educational Psychology Review, Vol. 10, No. 3*.

Tanenbaum, A. S., 2003. *Computernetzwerke.* München: Pearson Studium.

Walter de Gruyter GmbH & Co. KG, 2010. *Pschyrembel Klinisches Wörterbuch.* Berlin / New York: s.n.

Wijsman, J., Grundlehner, B., Penders, J. & Hermens, H., 2010. *Trapezius muscle EMG as predictor of mental stress.* New York, NY, USA, ACM, pp. 155-163. http://dx.doi.org/10.1145/1921081.1921100

Wittkowski, W. & Speckmann, E.-J., 2006. *Praxishandbuch Anatomie.* Erfstadt: area verlag gmbh.

Wu, H.-Y.et al., 2012. Eulerian video magnification for revealing subtle changes in the world. *ACM Trans. Graph.*, #jul#, 31(4), pp. 65:1--65:8. http://dx.doi.org/10.1145/2185520.2185561

Yin, B., Ruiz, N., Chen, F. & Khawaja, M. A., 2007. *Automatic cognitive load detection from speech features.* New York, NY, USA, ACM, pp. 249-255. http://dx.doi.org/10.1145/1324892.1324946

Yu, K., Epps, J. & Chen, F., 2011. *Cognitive load evaluation of handwriting using stroke-level features.* New York, NY, USA, ACM, pp. 423-426. http://dx.doi.org/10.1145/1943403.1943481

Abbildungsverzeichnis

Verzeichnis für Code-Listings

Formelverzeichnis

Stichwortverzeichnis

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.

Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.

Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.

Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.

Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

(Ort, Datum, Unterschrift)