Institut für Visualisierung und Interaktive Systeme

Abteilung Mensch-Computer-Interaktion

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Fachstudie Nr. 151

Evaluation der Brain-Computer Interfaces

Markus Funk
Hendrik Glück
Florian Pfleiderer

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Albrecht Schmidt
Betreuer:	Alireza Sahami
begonnen am:	01.01.12
beendet am:	10.05.12
CR-Klassifikation:	H.5.2 User Interfaces

Mediabrain
Video Annotation based on Brain-Computer-Interaction

Markus Funk, Hendrik Glück, and Florian Pfleiderer

Universität Stuttgart,
Institute of Human Computer Interaction
Fachstudie 'Media-Annotation' 2012
funkms@studi.informatik.uni-stuttgart.de
glueckhk@studi.informatik.uni-stuttgart.de
pfleidfn@studi.informatik.uni-stuttgart.de
http://www.vis.uni-stuttgart.de/institut/abteilungen/mensch-computer-interaktion.html

Abstract. In this paper we describe our 'Fachstudie' on the topic of BCI-based video annotation. During our work, we tried to resolve the question if today's brain-computer-interfaces could be a good solution for implicit multimedia annotation. We describe the BCI devices that currently are on the market and a introduce our prototype, Mediabrain. A user study to evaluate the results is being conducted.

Keywords: Emotiv EPOC, video annotation, Brain-Computer-Interfaces

Table of Contents

1 Introduction

Today, computers are omnipresent in everyone's daily life. A lot of time when handling simple machines, we are actually interacting with computers, not even realising it. But to make using those computers easy and possible for everyone, research in the field of human-computer-interaction is needed. Aware of this fact, we chose the institute of human-computer-interaction under the direction of Albrecht Schmidt for our study, hoping we could work in the fascinating field of interaction, away from classical mouse and keyboard. The institute gave us the opportunity to do exactly that and have a brief look at the quite new field of brain-computer devices. In this report, we will show our findings and introduce our software, Mediabrain, that is capable of media annotation based on brain input.

2 Course of the project

The work on our 'Fachstudie' can be divided into several phases. During the first phase, we did research on brain headsets in general. A good overview of brain-computer-interfaces that are currently on the market can be found on the

Wikipedia article "Comparison of consumer braincomputer interfaces". Several papers exist for most of them, describing their advantages and disadvantages. After having decided on which headset to use, the implementation phase started and we implemented Mediabrain, the software that is explained in detail in section 4. When the software was finished we conducted a user study to evaluate the functionality of our software. We used our findings to write a paper which we submitted to german "Mensch und Computer 2012" conference in Konstanz. In writing the paper we were assisted by Alireza Sahami, employee at the chair of human-computer-interaction and by head of chair, Prof. Albrecht Schmidt. The complete paper can be found in the appendix. The last phase was the writing of this very document, describing our work, our software and our findings.

2.1 Finding a topic

It took us a while to acquire the knowledge needed to determine which kind of program would be suitable to create. During the first period of our study, we did a lot of research on brain-computer-interfaces in general, trying to find out which ones would be a good decision to get a first look at the topic. We found out that there are already some quite interesting products on the market using BCI, for example an alarm clock that measures your brain activity during sleep to wake you up in a suitable moment. Focusing on BCI headsets, we found that Emotiv EPOC and Neurosky Mindset would be two good choices to have a closer look on, so buying those two was our recommendation to the institute.
Due to legal issues it took a while until we got the Mindset, so our program is based on EPOC to which we had access quite early during our 'Fachstudie'.
As you will see in a later section, the EPOC comes with three suites: facial expression suite, affective suite and cognitive suite. One time during the project it was in discussion to write one distinct program using each of these suites. Due to lack of sense and time, this was rejected later, but at this point we want to show the ideas which came up during that phase.
As for the facial expression suite, it was talked about implementing a program showing pictures of different emotions to mimic the face of the current user. But since basically the exact same program was already included in the package delivered with EPOC, we saw no sense in that. Later on, we thought it would be a cool idea to write a camera driver using the EPOC as input device, so you could for example wear the headset while doing a skype conference and having an avatar mirror your facial expressions without you being seen yourself. This concept was rejected due to lack of time and benefits. It was further on decided not to work with the facial expression suite and focus on the other suites.
The affective suite is the one that is used by Mediabrain, so one can already guess, Mediabrain was one of the ideas for what we could do with the affective suite. Other ones discussed were different types of media annotation like photos or music. We decided to use videos since you can look for single interesting scenes here while for pictures you could only rate the image as a whole (for songs it's usually the same, the interesting information is whether or not one likes a song, not which passages he likes best).

We also had some interesting ideas on how to use the cognitive suite, like a media (music) database you can navigate with your brain to choose the song you want to listen to. Some links with the annotation idea of the affective suite were talked about. But in the end, the concept was rejected since it wouldn't have offered anything special and be a whole lot of implementation work for the time period we had. Another idea was to implement a simple quiz (called "BrainBuzz") where the user could give his answer from a range of four possible answers by thinking of preconfigured A, B, C or D. We put some effort into that idea and we built a prototype of this cognitive based quiz game. This is described in detail in the future work section.

2.2 Related works

In this subsection we want to give a closer look at some works that we came across during our research and partly were just interesting and inspirational for our own work and could be helpful for anyone thinking about doing a project in the field of brain-computer-interaction.

Downloadable applications During our research, we found that it is possible to split the BCI applications available on the market today into several categories.

Games are a simple way to draw the attention of greater numbers of people on the field of brain computer interaction. How those actually work varies widely. The games on the Emotiv EPOC store for example are mostly very simple games (like pong, tetris or such) that can be entirely controlled via your mind. To do so, cognitive input is used. This of course is not possible for the Neurosky Mindset since it does not collect any cognitive data. This is why most of the Mindset games are controlled via affective input, additionally using classical input devices like mouse and keyboard. An interesting subsection of Mindset games would be educational games. The measures taken by those can help children to work out strengths and weaknesses and therefore optimize their learning curve.

Various controlling applications can be found for the EPOC, emulating input devices such as a mouse, which can be moved using your mind or keypress events on a keyboard. This can be a solution for handicapped people that would not be able to use a computational device otherwise. Another group that shall be attracted by this are gamers, since there are a lot of pre-made applications for popular games that let you brain-control specific actions like firing a gun in action games or casting magic spells in role playing games.

Also there is range of so called *meditation* or *atmospheric programs* that try to compute your current mood and change your screen according to it or that detect your current level of meditation to help you with meditation exercises.

Scientific works Beside the applications anyone can download from EPOC or Mindset store, there are a lot of scientific projects in the area of brain-computer-interaction.

Some focus on life-improvements for disabled people, making it possible to steer wheelchairs or control computers using your brain. Dasher [4] would be an example for a P300 based text input program that gives physically impaired people the chance to input text at a moderate speed.

Future application areas of BCI could include steering cars with your brain. At this time there are already some impressive videos around showing how this could work, but a closer look shows that those are autonomous cars and the only inputs they receive are the orders to turn left or right. Anyway, with some improvements in both of those sectors, controlling an autonomous car using your brain could be realistic in the future.
Another task that could be assisted by BCI and useful in a car is to initiate phone calls to your mobile contacts via brain input. Applications like ThinkContacts or P300 NeuroPhone allow to.

Studies conducted using BCI interfaces concentrate on topics such as the possibility to measure and categorize a users level of attention and meditation; collecting and assessing cognitive information from students while reading different texts; the ability to distinguish emotional responses reflected in different scalp when viewing pleasant and unpleasant pictures; analyzing the mental state of a car driver using a mobile phone while driving.
Brain signals are also used in medical context to diagnose brain disorders or help people with sleeping problems.

3 BCI-Interfaces

There are various methods to measure brain signals. The MEG (Magnetoencephalography) technique maps the brain activity by recording magnetic fields produced by electrical currents occurring naturally in the brain. fMRI (functional Magnetic Resonance Imaging) measures brain activity by detecting associated changes in blood flow. EEG (Electroencephalograms) measures brain voltage fluctuations resulting from ionic current flows within the neurons. It measures six signals each with different frequency ranges (alpha: 813 Hz, beta: 1330 Hz, gamma: 30100+ Hz, delta: up to 4 Hz, theta: 48 Hz, mu: 813 Hz). With the advantage in new technologies, commercial EEG headsets are recently available and can be used for laboratory studies.

During our work on the subject we had the chance to lay hands on two brain-computer devices: Emotiv EPOC and Neurosky Mindset. This section will have a closer look at their capabilities.

3.1 Emotiv EPOC

The Emotiv EPOC is a quite powerful device, using as much as 14 sensors to measure brain activity. This gives the possibility of measuring a lot more data than you can with the Mindset and it's four sensors. Emotiv comes with as much

as three suites, facial expression suite, affective suite and cognitive suite, each delivering a lot of different data.

Facial expression suite works by measuring the currents that muscular activity in your face produces and can therefore quite accurately (after some training) tell what your face looks like at the moment.
Affective suite gathers data about excitement, boredom, affection, frustration, engagement and long term excitement. For all of these data it delivers a value between 0 and 1 to give you a procentual measure.
Cognitive suite allows you to train and recognize up to 14 different actions, 4 of which can be active at the same time and therefore be triggered by your thoughts.
The EPOC store currently offers a small variety of applications designed for the headset, but still developing your own applications using the SDK would be the most interesting aspect of EPOC right now.
One of the last barriers which prevents a widespread acceptance by the industry could be the price for one headset. At the moment, each device costs around $300. Although this is a price that educational institutes can easily afford, it still is quite a lot for private users. Emotiv offers a variety of versions depending on the background of the developer. The prices for the SDK range from $500 for independent developers and small firms to $2500 for the educational and enterprise versions [3]. At the moment, the SDK comes with support for the languages C++, Visual Basic (.NET) and Java [3]. Furthermore, it names C# as being supported as well, but we were not able to compile the C# wrapper provided by the SDK. Research into this problem revealed that there are currently bugs (most probably related to Windows XP operating systems) which lead to our decision to implement our projects using C++ instead of C#. Issues with different platforms seem to be a general problem for the Emotiv SDK since it is currently only available under Windows with Mac OS and Linux being listed as "coming soon" on the Emotiv homepage [3].

3.2 Neurosky Mindset

The Neurosky Mindset is a lightweight solution suitable for the private market since it is just a normal audio headset, amended by a single sensor on the forehead and three sensors on one ear. It can be easily paired to a lot of devices (desktop, laptop, mobile) via bluetooth. In contrast to the rich data provided by EPOC, the Mindset is only able to present you two values: concentration and meditation. Still there are lots of applications (as much as 70 on the Neurosky store) available for the Mindset and Neurosky collaborate with 300 universities around the world to develop new hard- and software for BCI and Mindset. To attract researchers they grant easy access to development and research tools.

4 Mediabrain

Mediabrain is a software that allows video annotation based on brain input. It is written in Visual C++ and uses the Emotiv EPOC headset and Emotiv SDK,

VLC media player and Xerces XML Parser. The motivation to implement this software was to find out if brain data can be used to automatically determine highlights from a video without the need of any explicit input by the viewer. One possible field of application could be finding the interesting moments of a sports event. Could it be possible to detect the highlights of a football game by letting a number of people watch it while wearing a brain headset? Currently Mediabrain only uses the viewer's excitement value to detect possible highlights. The use of other values like engagement or frustration could be a possible extension in the future.

The program can be divided into two separate parts: gathering of data and evaluating of the collected data. Since the Emotive SDK only offers functions and datatypes in C, we had to implement a C++ wrapper to be able to write our program in an object-oriented language. As a solution for persistent data storage, we decided to use xml files. We defined an xml schema for collected brain data that can be used for evaluation of any kind. As for video playback we decided to use VLC media player which supports easy embedding and is capable of playing almost any video format.

4.1 General Architecture

We built Mediabrain with the Model-View-Controller paradigm in mind. As you can see in figure 1, we constructed three architectural layers. At first, we created a Model which holds all input that comes from the Emotiv EPOC headset. The second layer uses the data that is collected from the headset and puts it into XML-files. This Application-Layer also controls the Presentation-Layer and manages to correlate the brain-data to the timestamp that comes from the watched video. The third layer simply wrappers around the VLC Media Player and controls the video playback to be synchronous with the recording of the data.
All layers are described in detail in the following sections.

4.2 Component GUI

The GUI component uses the VLC Media Player to playback videos. For the integration we used the examples from [1]. This wrapper basically provides commands to send events like start/pause/mute to the VLC-Player. We built a GUI using MFC that contains of 3 frames.

The main-frame (see figure 2) is intended to provide a live-overview of the data that comes from the Emotiv EPOC. When doing a study with Mediabrain, the tester can place the mainframe on a separate monitor to see live data that the participant can't see. Using the menu, the user can open the other frames of Mediabrain.

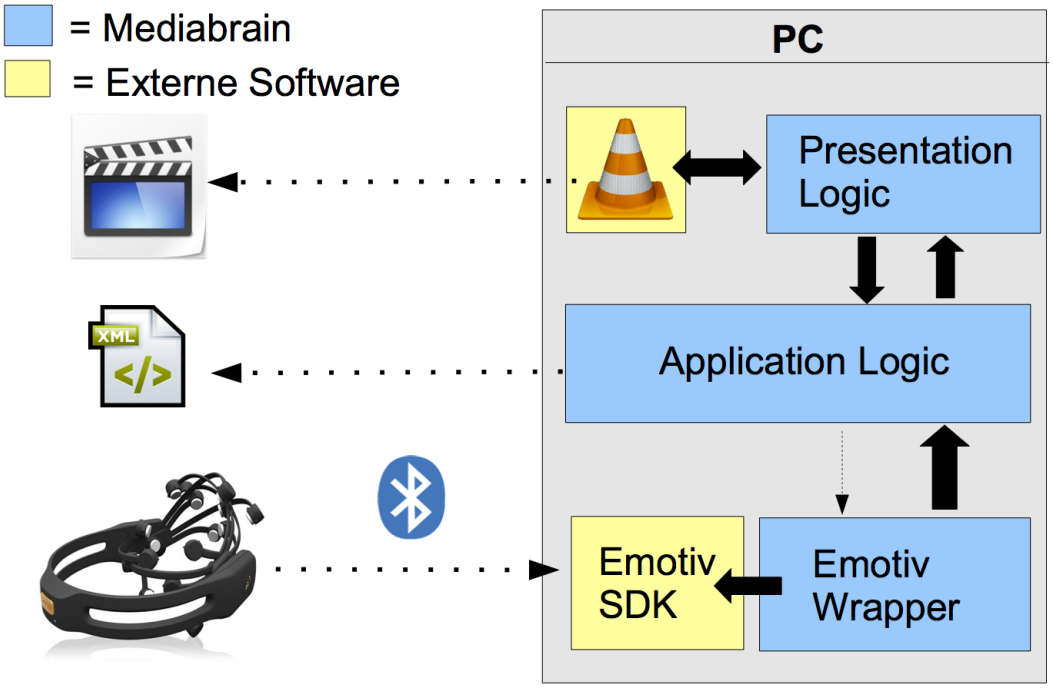

Fig. 1. Overview of the general architecture

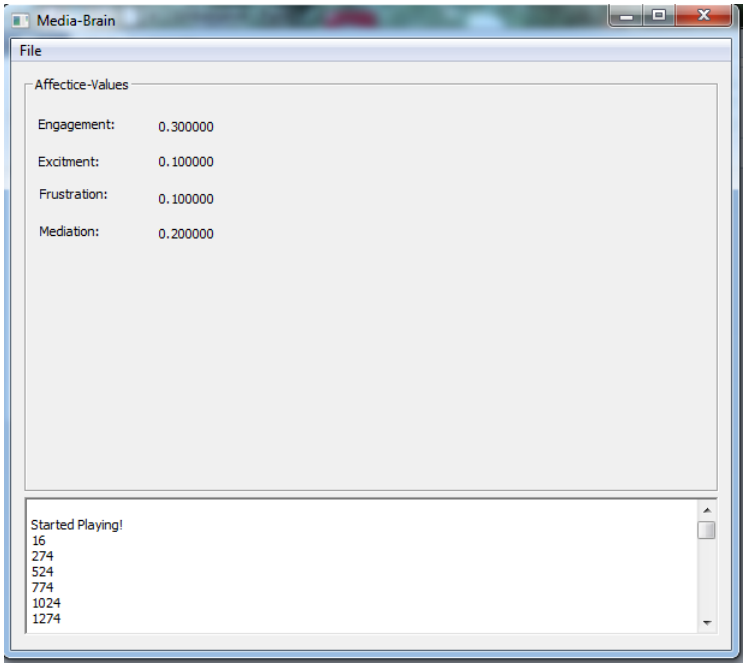

Fig. 2. The mainframe of Mediabrain

The recording frame (see figure 3) basically looks like the VLC media player and is completely integrated into Mediabrain, permanently logging the video's timestamp and matching brain data. As in the real VLC media player, the user can start, stop and pause the video. Also the user can mute and control the volume. When the user presses the stop button, a save routine is being invoked. This save routine writes all brain data correlated to the video timestamp into an XML-file.

The third view (see figure 4) is used to calculate the highlights of the video using the recorded data. The highlight view can be opened via the file menu of the mainframe. At first, the user has to select an existing XML file which is then parsed. In the GUI of the highlight view, the user can set two parameters. The first is the number of highlights that are to be calculated. The second parameter is the length of a highlight. Varying those parameters can result in better calculation of the highlights (as described in 4.8). When clicking on the calulate button, the calculated highlights are being displayed in the list. The user can watch the highlights again by selecting a highlight in the list an hitting the play

Fig. 3. The recording frame of Mediabrain

button.

For conducting the user study, we built an additional view where the user can explicitly define highlights while watching the movie (see figure 5). This view just consists of a single button, so the researcher can decide whether the user should define the highlight via the enter button or a mouse click. In the background, the current timestamp of the video is being recorded and can be saved to a file for further analysis.

4.3 Implementation

Mediabrain is realised as a Visual C++ application using the GUI framework provided by the Windows SDK and developed with Microsoft Visual Studio 2010. It mainly consists of a set of dialog classes used to display the user interface as well as various data and utility classes. The latter provide functionality for persisting the data (XMLUtils) as well as evaluating the input from the Emotiv device (EvaluationUtils). Furthermore, as aforementioned, we include a VLC wrapper which allows the application to play a broad variety of video formats for annotation. Instead of directly using the Emotiv SDK, we use a wrapper which is described in detail in the subsection "Component SDK-Wrapper". The application can assume different states:

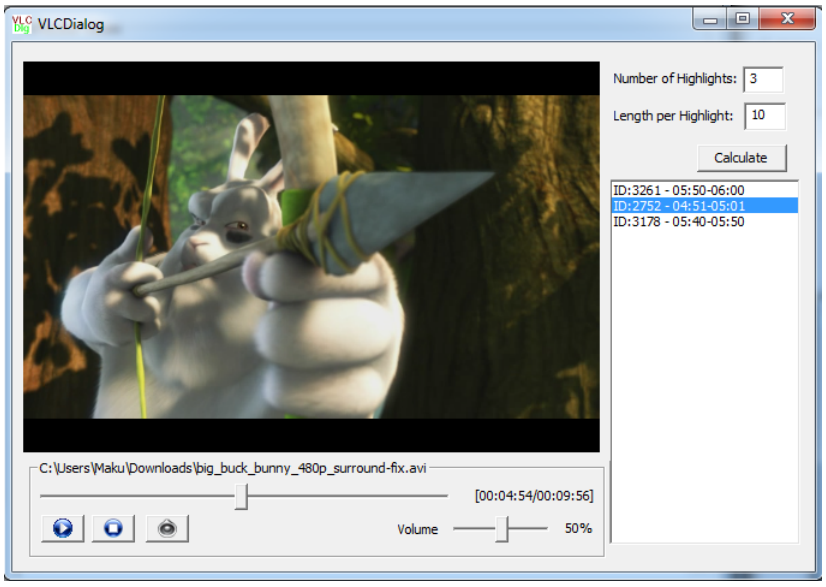

Fig. 4. The highlight-view of Mediabrain

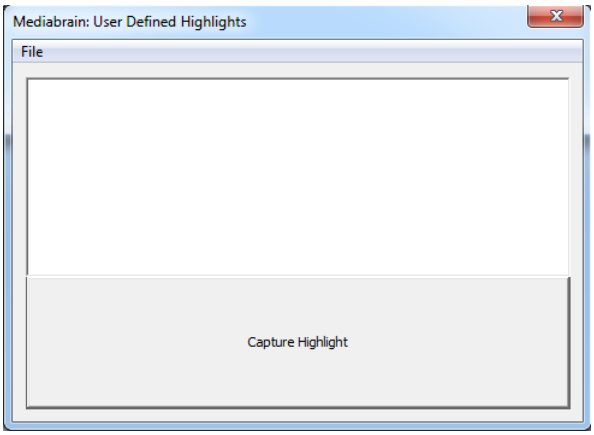

Fig. 5. The user defined highlight view of Mediabrain

- **Idle:** In this state, the application is waiting for the user to select an option which leads to one of the following states.
- **Recording:** During the recording state, the application is playing a video selected by the user. Meanwhile, the EmotionRecorder is storing all the necessary information provided by the Emotiv headset. When the video stops, all data gathered during the recording is persisted into an xml file.
- **Highlight Playback:** The last state of the application is assumed when opening an annotation xml file. Mediabrain displays the highlight dialog and waits for the user to select a highlight for playback.

4.4 Component SDK-Wrapper

The first thing we noticed when we started to work with the Emotiv EPOC SDK was the fact that it, although it is stated as being written in C++, was rather a C implementation. Considering the advantages of a class-based concept, we decided to write a sophisticated wrapper for the functionality provided by the SDK. The following items describe the key aspects which were realized by our wrapper:

- Logical structure for methods and fields.
- Multicast architecture for dispatching events.
- Reusable code for minimized effort
- Simplified methods for cognitive and expressive training.
- Asynchronous, thread-based pulling of Emotiv state information.

The first aspect was achieved by introducing a class called EmoEngine which acts as a central controller for the communication with the EPOC headset. With this class it was possible to aggregate a lot of functionality which is spread across several separate functions and data types in the SDK. Among other things, the EmoEngine class provides methods for simplified headset connection management, training procedures for the cognitive and expressive suites as well as a parallel thread which pulls state updates at a given interval.
Furthermore, the EmoEngine realizes the *Observer Pattern* which means that it allows for so called *Listeners* to be registered to it and later be informed about occurring events like establishing a connection, fetching a status update or completing cognitiv training. Doing so allowed the EmoEngine to dispatch those events to multiple listeners in a multicast way.

EmoEngine The EmoEngine class is the actual core of the component SDK-Wrapper. As mentioned in the last paragraph, it aggregates various functions which are spread across the Emotiv SDK. In this subsection, we are going to take a more detailed look at how the class is structured, what interfaces it provides and how the functionality is realized. The structure of the EmoEngine class is actually quite simple. For usage, it has to be instantiated. After that, an application using this class can register so-called event listeners (or handlers) which will

inform it about upcoming events. This process is described more detailed in the subsections following this paragraph. For this part, it is sufficient to know that the EmoEngine stores a list of all registered event listeners so that they can be notified once an event occurs. The notification can be done using a set of provided methods which start with the prefix *fireOn*. An example would be the fireOnConnect-method which can be used to inform all registered listeners about an established connection to the Emotiv headset. Those methods are declared in the protected scope which allows them to be used in case future projects want to derive from the EmoEngine class. Figure 6 shows how the class is structured.

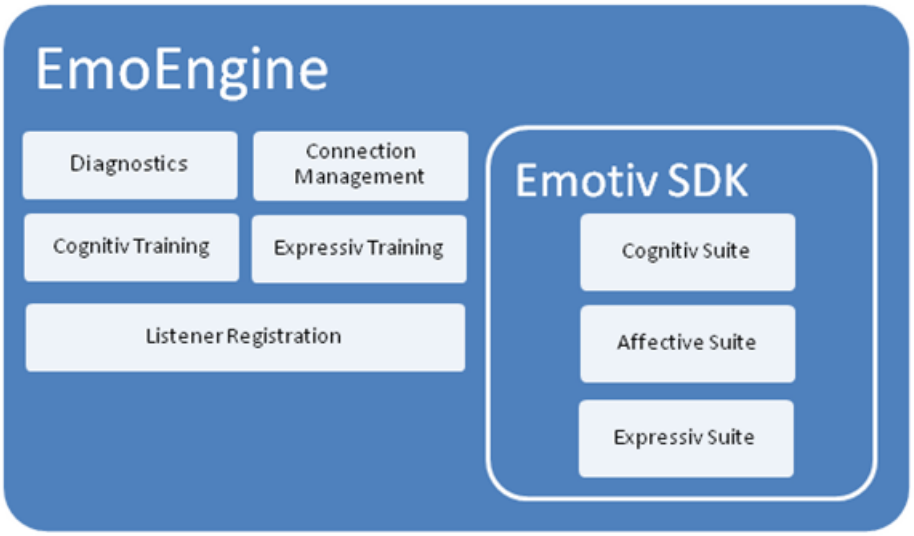

Fig. 6. Structure of the EmoEngine class.

Apart from the storing of the listeners, member variables exist for representing the current state of connection. This information is split into two parts as the EmoEngine can be connected in two different ways: Locally, which means that it is connected to an Emotiv headset which is directly coupled with the computer running the engine, and remotely, which allows the engine to connect to a remote device using it's IP address. Remote connections are for example used to connect to the EmoComposer, a tool provided by the Emotiv SDK. It emulates a headset and can therefore be used to debug applications using the SDK. Note that an instance of the EmoEngine class can only be connected to exactly one source of data. If the source should be changed, then the engine has

to disconnect before it can establish a new connection to a different target.

The next part of the structure which should be described here is the logging functionality. The Emotiv SDK provides an interface for enabling a logging system called *diagnostics*. The EmoEngine provides methods for starting and stopping the logging of information from the SDK. To start the diagnostics, the application has to provide the path to a logfile into which the output will be written. Note that this is normally only used in the debugging process and should be disabled when releasing the application.

Eventually the last part of the EmoEngine which is specified here are the structures responsible for storing information about the current training state of the user. These members are only visible to the class itself as they are needed to tell, in case of an incoming training event, if the application is currently training a certain action. Furthermore, it stores a flag that indicates if the engine is waiting for the application (or the user) to make a decision concerning a completed training. These actions include accepting or rejecting the results of a training process. A more detailed description on how the training is done can be found in the subsection "EmoExpressivEventHandler". The EmoEngine class provides a set of methods for starting a training as well as accepting or rejecting the results of a completed training sequence. Furthermore, it offers the possibility to check if a training sequence is currently in progress. All mentioned training methods are implemented for cognitive as well as expressive training.

One way of realizing the aforementioned event listeners would have been to specify a single class which would define all the possible callback methods. Based on the fact that most applications wouldn't care about most events, we decided to split them up into separate listener classes. This lead to the inheritance hierarchy seen in figure 7. The following subsections describe the components of the event handler hierarchy.

BaseEventHandler: This is the base class for all event handlers. The reason for introducing this class was to have the possibility to add methods or fields which should be shared by all descendants. At this point of time, no such members are present or needed.

EmoConnectionEventHandler: This handler must be implemented and registered at the EmoEngine if an application needs to be informed about changes in the connection state of the SDK. Therefore, it provides two event callbacks: *onConnect*, which is called when a connection is successfully established to the EPOC headset and *onDisconnect*, a callback that informs the application about a closed connection.

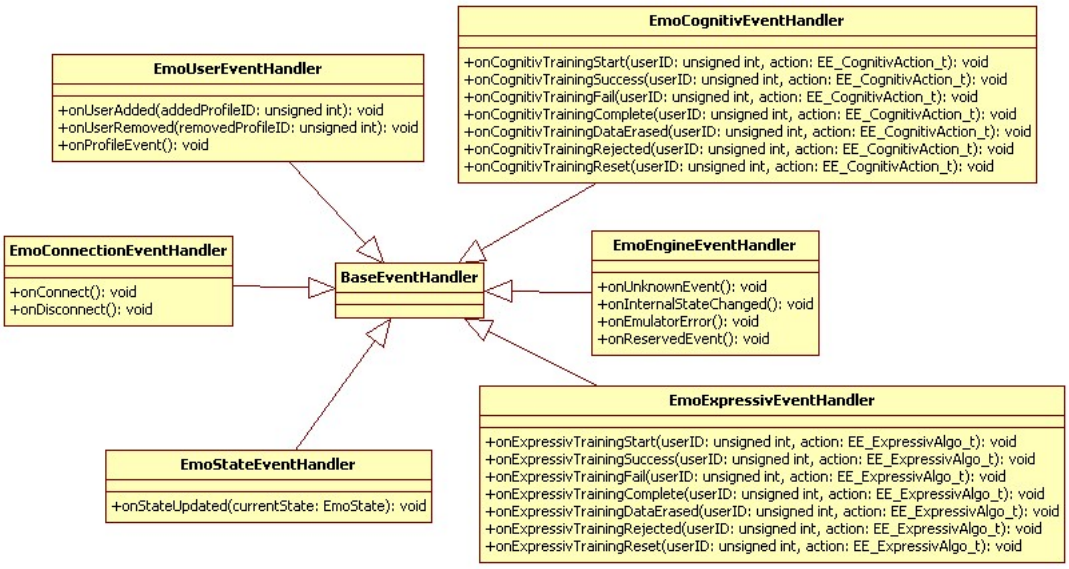

Fig. 7. Inheritance of the event handlers.

EmoUserEventHandler: To be able to handle multiple persons using one EPOC headset, an application which is working with the Emotiv SDK must provide a mechanism for managing multiple user profiles. Those profiles would be needed to store for example the training information which allow the headset algorithms to be adapted to the individual brain activities of the users. The EmoUserEventHandler provides methods which will be invoked once a new profile is created at or removed from the headset.

EmoStateEventHandler: The EmoStateEventHandler is very-likely the most important event handler. It will be triggered every time the background thread fetches new data from the headset. Almost every application will need at least one of those handlers to retrieve the information provided by the expressive, affective or cognitive suites. Data received this way is aggregated by the EmoState class. Normally it, if working directly with the SDK, it would be necessary to fetch all the information manually by requesting them from the headset. Our wrapper takes care of this with the aforementioned EmoState class. In addition to that, the EmoState provides utility methods for testing if the data generated by one of the suites has changed since the last update. More detailed information about how to work with EmoState can be found in section EmoState.

EmoExpressivEventHandler: This handler is one of two handler classes which deal with training events. The expressive suite of the Emotiv SDK provides information about the facial expressions of the user like smiling, winking, and direction he is looking at. As the detection of those values differs for every user, the algorithm has to be trained in order to produce reasonable results. Training mechanisms have been implemented into our wrapper and an EmoExpressivEventHandler is needed in order to be informed about the state and result of the training. Furthermore, the user should be given the possibility to reject or accept a training session. Figure 8 shows the sequence of a successful training session. Note that the application accepts the training in this case. It would also be possible to reject the training if the user felt that his expressions would not suffice for training information.

EmoCognitivEventHandler: The EmoCognitivEventHandler is very similar to the EmoExpressivEventHandler described in the last section. Its purpose is also to deal with training sequences. One of the most exciting functionalities of the Emotiv SDK is the detection of cognitiv actions which can be mapped to the patterns described in the Emotiv EPOC section. As the detection of these actions relies heavily on them being trained before, the EmoEngine wrapper has built in support for doing so. The sequence of the training is similar to the process described in figure 8.

4.5 Component Evaluation

When thinking about how to evaluate the collected data, we decided to focus on the excitement value. The user's excitement seemed to adequately represent his tension and affection to the movie.
Our algorithm works in three steps:

1. sorting
2. filtering
3. highlight generation

In the **sorting** phase, all events that have been captured in the data structure (one for every second the user watched the video), are ordered by their excitement values. So the data now is in a state where we can tell the most exciting moments are in the first array elements while less exciting ones are in the back. The **filtering** phase now helps to sort out what we are going to call "double highlights". Since we got values for each second and real highlights are going to last longer than that, we will have several high measurements for the same highlight. This could lead to conflicts if we only returned the n measurements with the highest excitement value. By ensuring a certain time delta between highlights, the filtering phase allows us to return a single highlight only once.

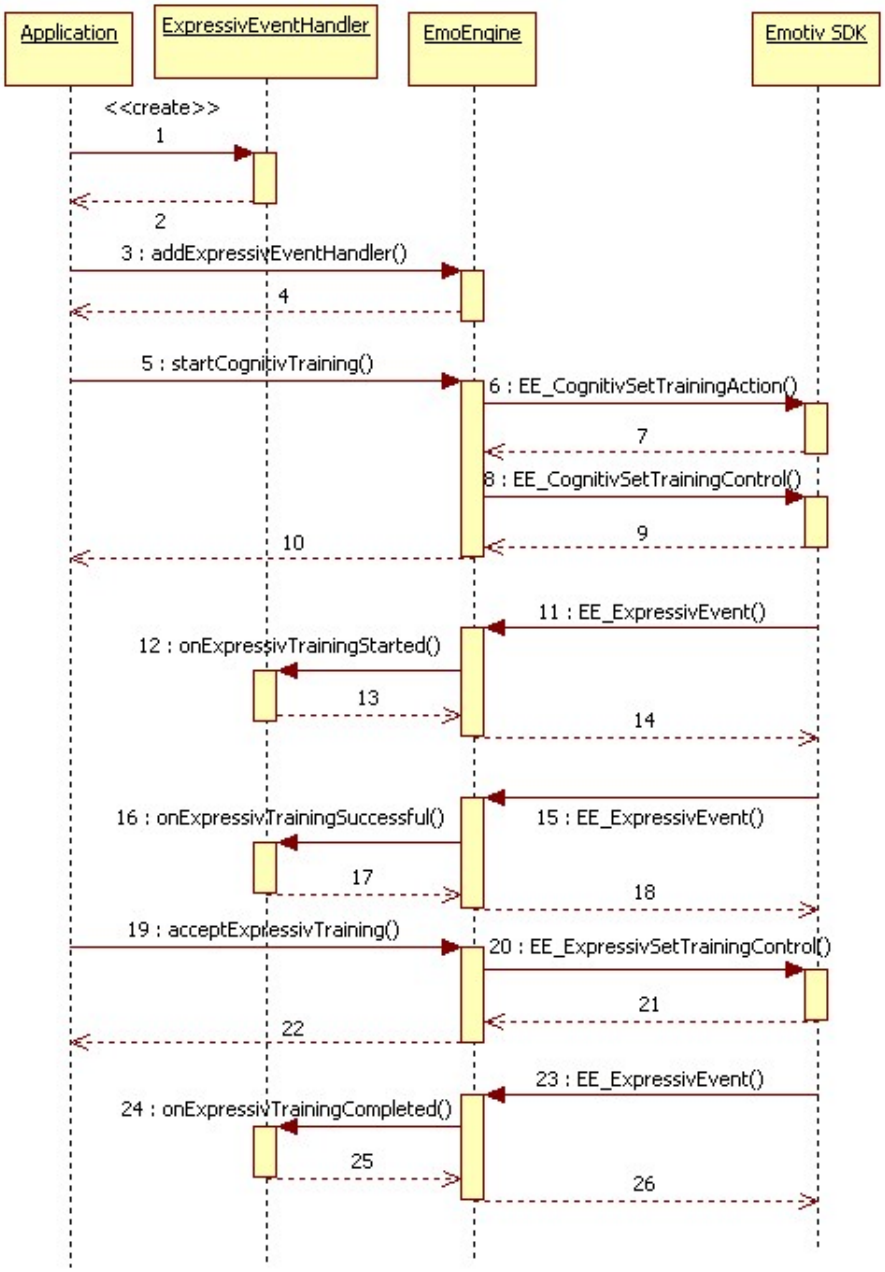

Fig. 8. Process of an expressiv training.

In the **highlight generation** phase, the now found highlights are converted into a format usable for our software and actually returned so they can be shown in the evaluation component.

During our user study, it came out this algorithm is not optimal to actually find user highlights because of the way the human brain behaves and the data Mediabrain can get. We expected the excitement value to immediately go up when the user is excited. But it turned out that very moment the curve just starts to rise and reaches its maximum a lot later. Therefore in a second iteration the algorithm should be altered so that it can detect changes in the gradient of the excitement function.

4.6 Component Datastructure

The data structure used to store our collected data is xml-based. A valid Mediabrain xml file has to fit the xsd-specification given in figure 9

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="VideoData">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="document" type="xs:string"/>
      <xs:element name="user" type="xs:string"/>
      <xs:element name="recordingtime" type="xs:integer"/>
      <xs:element name="datachunk" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:string"/>
            <xs:element name="timestamp" type="xs:string"/>
            <xs:element name="excitement" type="xs:decimal"/>
            <xs:element name="engagement" type="xs:decimal"/>
            <xs:element name="meditation" type="xs:decimal"/>
            <xs:element name="frustration" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Fig. 9. The xsd specification of Mediabrain save files

As can be seen, we collect much more data than we currently need to evaluate user highlights. This is because several extension to highlight determination could be possible in the future, some of which are:

- using other values (e.g. affection or long term excitement) in the highlight calculation
- using measured data to manually determine coincidences for highlight algorithm improvement
- using facial expressions as an indicator for user mood

For the latter, the xsd schema would have to be extended to hold facial expression data. Because of the simple format used and the easy extendability of xml this could be done quite simple. Using the cognitive suite has not been considered so far.

The writing of xml-files is done using the c++ xerces DOM parser, which is a light-weight, easy-to-use solution. The code fragments using xerces were written as understandable and well-commented as possible to ensure easy extendability. During the recording of data, all data is stored in a program-specific data structure (see figure 10)and only when saving a DOM tree is generated to write data. Vice versa, when performing evaluation, the data is read from an xml file and then converted into this program-specific structure before any evaluation tasks are performed on it.

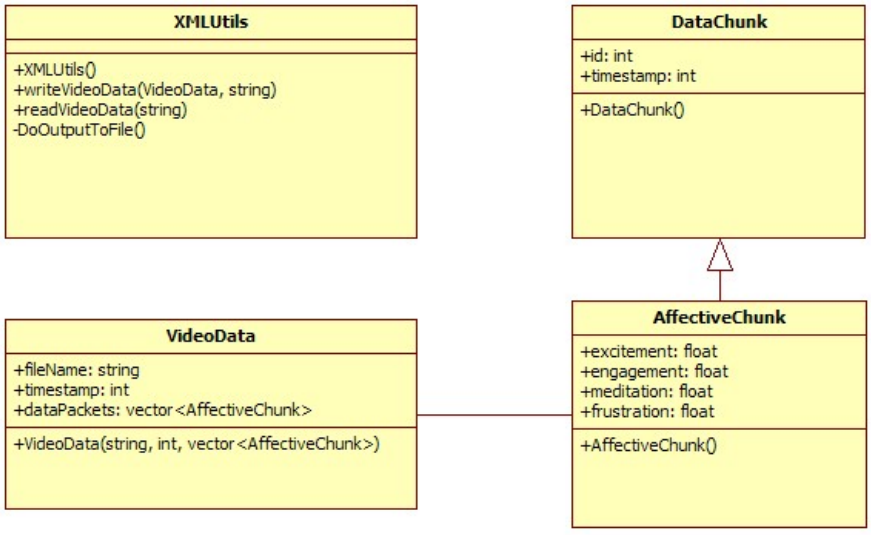

Fig. 10. The internal data structure used in Mediabrain

4.7 Further Usage

Based on the fact that one component of our work was a generic wrapper for the Emotiv SDK, the possibilities for further usage are manifold. With the wrapper it is possible to write further applications with a reduced amount of code needed to include the SDK. Furthermore it takes care of the necessity to implement a way of asynchronously fetching state updates from the headset. Apart from Mediabrain, our wrapper has already been deployed to several other projects by other students. Among those projects is a connector which allows to broadcast the data provided by the wrapper to multiple receivers using the UDP transmission protocol. Using this technique it was also possible to integrate the Emotiv headset as an input device into the EIToolkit, a framework developed by the HCI institute which connects all kinds of different input devices (for example, eye trackers) using UDP based commmunication.

With UDP being platform and language independent, our work could also be used in other languages apart from C++. A project in which this is being done is currently developed by Markus Funk. He uses the UDP transmitter to send signals provided by our wrapper to a game client written in Java. In that game the user controls a basket with his thoughts. For this, three actions are defined: Left, Right and Neutral. The user now collects things that drop from the top of the screen. Points are given according to the number of collected things.

One downside, which limits the uses of our work is the fact that the threading routine used to fetch the data is based on the WINAPI provided by Microsoft Windows. Therefore, although porting would be not too difficult, our wrapper is currently only usable in Windows-based environments. These limitations have been weakened by the UDP wrapper which allows to transmit the information to other operating systems like Linux or MacOS.

Another field of possibilities arises from the Mediabrain application. Part of Mediabrain is a so-called *EmotionRecorder*. This component is responsible for the aggregation and persistence of the affective data received from the headset. While bearing in mind that further information could be useful later on, we implemented the recorder to store additional data apart from what our highlight detection algorithm actually uses. Therefore, Mediabrain, especially the *EmotionRecorder* component, could be used in new projects which analyze the affective state of the user. Re-usage of Mediabrains data is even more simplified by the fact that they are stored in an easy-to-parse XML structure.

The last project to be presented in this section is "BrainBuzz", which was already mentioned earlier in this document. BrainBuzz was inspired by the popular quiz game Buzz [5] which is available on Sony Playstation 2 and Playstation 3. Buzz allows the player to take part in a quiz similar to what can be seen on TV in a lot of different shows. Up to 8 players can compete against each other by answering questions using a device called "buzzer" to select the correct answer from 4 given options labeled A to D. BrainBuzz takes this concept and intro-

duces the Emotiv headset as a substitute for the buzzer input device. Since it is (currently) impossible to detect if the user is thinking of the correct answer, another approach had to be taken to detect which answer should be locked in as a users choice. The feature suited best for realising this is the cognitiv suite provided by the Emotiv SDK. Therefore, the four different answers are mapped to four of the cognitive actions (currently: Pull, Push, Right and Left). As mentioned in this paper, detection of cognitive actions requires training them to achieve acceptable recognition rates. Therefore, before playing, the players have to train each option. Another difficulty that had to be considered was the fact that a user could accidentally select an answer by unintentionally thinking of an option. To allow switching their selection, the game checks if the cognitive action value is higher than a certain constant value. This value is different depending on the state in which the application is. If no answer is selected, the value is at 0,5. Once an answer is selected, a countdown is started. While the countdown is running, a reselection is possible by thinking of another answer. The value needed to switch an existing choice is higher than the one for the first selection. When the countdown is finished, the selection made by the user is checked for its correctness.

At the time this paper was written, the BrainBuzz game was implemented to a point where training and playing was possible but no graphical user interface was provided.

4.8 Additional Information

This section describes the experiences we made while using Mediabrain.
One thing we had to discover that the brain headsets we used do deliver good data as long as they are connected, but unfortunately the connection tends to be unstable sometimes. This leads to the generation of lots of unusable data. To fix this problem, the headset has to be put near the USB dongle until reconnected. A good workaround we found for this is using a USB extension cord and putting the dongle on the table near the user. This way we could keep up a steady connection.
Furthermore, it can take quite a lot of time until the headset finds any brain input at all. It was not unusual that users needed to wear the headset for several minutes until it actually recognized being on somebody's head. Even though information gathered from that point on is quite good and reliable, this leads to a high degree of user frustration and displeasure. We recognized full hair as being a potentially obstructive factor that can make it impossible for the headset to discover any brain activity.

For the calculation of the highlights the user has to set two parameters, the number of highlights and the length per highlight. We made good experiences in calculating 5 highlights with a 10 seconds length for a 10 minute movie. We recommend not to go below 10 seconds per highlight since the user needs to have at least 5 seconds before and after the highlight moment.

Unfortunately the current version of Mediabrain contains a bug that we were not able to fix in time. When manually aborting the recording of brain data

while watching a movie, it might happen that the logging mechanism and the stop routine try to write into the same vector at the same time, which results in a crash. To avoid this error, we recommend to watch the movie until the end where the save routine is being triggered automatically. The most probable reason for this is the fact that the wrapper uses a separate thread for asynchronous logging. Therefore, to fix this problem, a synchronization mechanism would be needed.

5 User study

For evaluating Mediabrain, we conducted a user study. The goal of the study was to find out whether the highlights that are calculated by Mediabrain are accurate and match the user's subjective impression or not. Therefore we invited 17 probands to participate in the 30 minute long user study.

5.1 Setup

The study was divided into two parts: the implicit and the explicit interaction.

The implicit interaction was the main part of the study. We made the participants wear the Emotiv EPOC and made sure that all sensors had a good connectivity (green light in the EPOC Control Panel). After that, we explained to the participants that the headset is recording their excitement while watching a movie. We also asked them to press the Enter-Button on a keyboard every time they feel excited about a scene in the movie. (We did not tell them that they trigger a user-defined highlight). After telling them about their task, we let them watch the movie Big Buck Bunny [2] (see figure 11). We chose this movie, because it has some distinct funny and exciting scenes that can be seen as a highlight. Choosing a movie with a constant level of excitement, like for example a fail-compilation, would result in an inaccurate highlight calculation. If we chose such a movie, we would not have been able to distinguish between a highlight and the normal state of the user since highlights happen to often. After letting the users watch the movie, they had to fill out a form that asked questions about wearing a brain-computer-interface.

The second part was the explicit interaction with the Emotive EPOC. For this, we used the cognitive suite demo of the Emotiv EPOC Control Panel. In this demo (see figure 12), a cube is hovering in the middle of the screen and the user has to use his thoughts to move the cube in the three-dimensional space. As the cognitive suite works like a pattern recognition algorithm, the user has to record a certain baseline to tell the system what his mind is like when being in a neutral state. After that, the user has to train the action that he wants to perform. Available actions are, for example, Push, Pull, Rotate left, Rotate right and many more. For the user study we let the participants train the actions Neutral, Push and Pull. After a 5 minute tryout period, we gave the user the

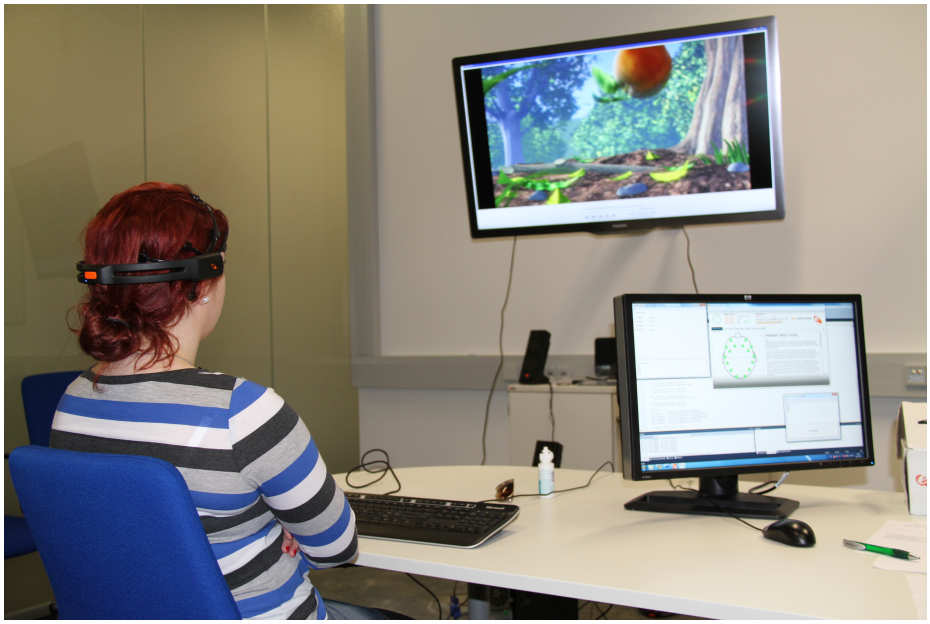

Fig. 11. Participant taking part in the userstudy

task to perform the actions that the conductor of the study told him to. The user had 5 seconds to perform the action. After that, the user had to go back to a neutral state. Then the conductor continued calling actions. We gave each participant a sequence of 8 actions consisting of Push and Pull.

6 Evaluation

In the implicit part of the study, we realized that the highlights calculated by Mediabrain were not as satisfying as expected. The users reported that they could not tell for sure if the calculated highlights fit their expectations. In the analysis of the user-defined highlight data, which the user had to trigger while watching the movie, we found out that the highlights computed by Mediabrain and the highlights defined by the user differ. We discovered that the user-defined highlights are not, as expected, the maxima of the highlight graph. If you take a look at figure 13, you can see that the highlights are located rather at the local minima than at the maxima. This is an interesting finding, because the excitement values that come from the Emotiv EPOC seem to be not as accurate as expected. Our analysis showed, that the moment which triggers a change in the gradient of the excitement curve can be considered a highlight. All further excitement values are just responses that are affected by the triggering moment.

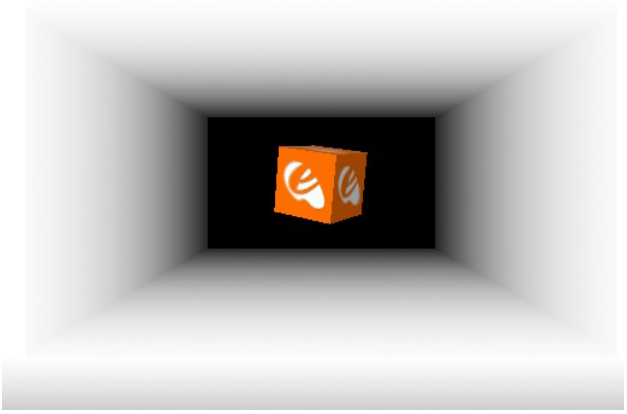

Fig. 12. Screenshot of the cube from the cognitiv suite

Whithin this study, we cannot tell whether this delay comes from the Emotiv EPOC or the human body. In further work, one could try to find out, how long the Emotiv EPOC takes to react upon exciting situations and how those situations are being reflected by the signals that come from the Emotiv EPOC.

The second part of the study showed the usability and the accuracy of the cognitive suite. Overall, the 14 users could trigger a specific action with 56.25% success, but the success rate differs between individual users. One user could trigger all 8 actions without any errors and therefore had a success rate of 100%. Other users only had success rates of 37.5%. The subjective opinion of the users was ambivalent. At first the users were impressed by the technology. They told us, they only knew such a technology from science fiction movies. And actually controlling a PC using only their thoughts was quite impressive. Nevertheless, their enthusiasm was strongly lowered by the huge number of errors in the detection of the actions they were thinking about. Users told us they never felt in control while using the cognitive suite of the Emotive EPOC. This is explainable regarding the high error rate of 43.75%. In further research, one could try and find out what such an experiment looks like with higher practice time. Furthermore, one could do long term study regarding the learning effect while using the the EPOC's cognitive suite over several weeks.

7 Conclusion

The research showed that the affective suite of the Emotiv EPOC has a certain delay regarding the actual trigger of an event and the maxima in the excitement graph. This shows that an implicit highlight recording should recognize a change in the gradient of the graph rather than its maxima. We suggested an algorithm

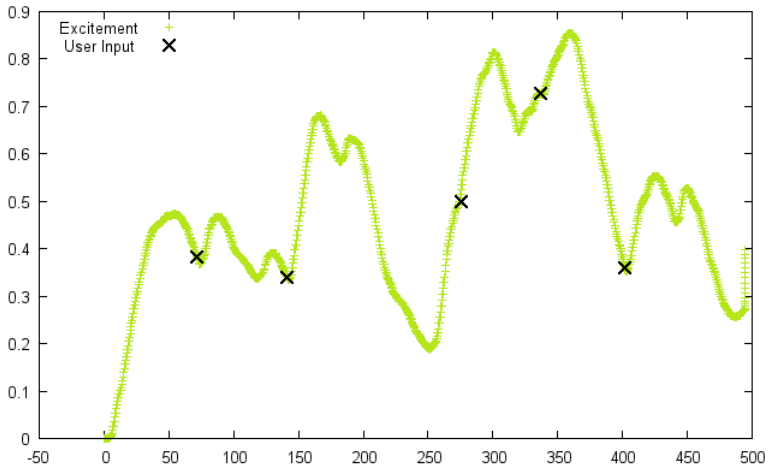

Fig. 13. The excitement graph of a user

that takes this fact into account (see also in our paper in the appendix). In future implicit video annotating applications, the other affective and additional expressive values(e.g. laughing) could be included in the calculation of the highlights. This should provide an even more precise detection of the highlights.

Also, our study shows that it is possible to implicitly annotate video files while watching them. This opens a whole new perspective for building applications. For example crowd-sourced highlights of a soccer game could be calculated with the input of a few users. Another idea would be to let media research groups test new movies and analyse how well the crowd reacts to them.

Our subjective impression of the Emotiv EPOC's cognitive suite is, that it is not yet suitable for critical application areas like driving a car using brain input. Possibly, the use within an application area where a high error rate is tolerable, the cognitive explicit input via the Emotiv EPOC is achievable.

References

1. VLCWrapper - Alex Skoruppa `http://www.codeproject.com/Articles/38952/VLCWrapper-A-Little-C-wrapper-Around-libvlc`
2. Big Buck Bunny `http://www.bigbuckbunny.org`
3. Emotiv `http://www.emotiv.com/`
4. *Text-input using a brain-computer interface: introducing Dasher*; van Vliet, M.; 2007
5. Buzz!^{TM}`http://www.buzzthegame.com`

Appendix

- *consent form for the user study* Each participant in the study had to read and sign this form.
- *checklist for conducting the userstudy* This is our personal checklist to make sure the user study takes the same process for each participant
- *list of actions to perform for participants* Each participants had to perform these actions in the given order during the "explicit brain control" phase of our study
- *final version of the paper* The final version of our paper as submitted for the MuC2012

Einverständniserklärung zur Studie
Media-Brain
„Implizite vs. Explizite Interaktion mit Brain-Computer-Interfaces“

Teilnehmer-Nr.: _____ Name: _____

Vorname: _____

Mir wurde der Zweck der Studie erläutert und ich wurde darüber informiert, dass ich die Studie jederzeit abbrechen kann.
Ich wurde des Weiteren darüber in Kenntnis gesetzt, dass die während der Studie gesammelten Video- und Audiodaten zur Auswertung der Studie herangezogen werden. Sämtliche Daten werden für die wissenschaftliche Nutzung gesammelt und hierbei vertraulich behandelt.
Der Verwendung von Bildmaterial mit meiner Person in Präsentationen, Ausarbeitungen und Veröffentlichungen (auch online)

[ ] stimme ich zu.
[ ] stimme ich nicht zu.

Stuttgart, den _____

Unterschrift: _____

Walkthrough – Studie: Media-Brain

1. Einverständniserklärung ausfüllen lassen
   - Zweck erklären (Fachstudie / Evaluierung von System / Paper)
   - Jederzeit abbrechbar
   - Bild/Videomaterial wird aufgezeichnet

2. Eingangsfragebogen ausfüllen lassen

3. Headset anschließen

4. Neuen User im SDK anlegen

5. Media Brain starten
   - Run User Highlights
   - Brain Werte checken
   - Run Video Demo

6. Speichern von VLC und Userhighlights
   Format: <Teilnehmernummer>_<vlc/user>.<txt/xml>

7. Zwischenfragebogen implizite Interaktion

8. Check Signal (noch alles ok ?)

9. SDK - Cognitiv Suite erklären

10. Train Neutral Action
    - Auf weiße Wand konzentrieren

11. Train Push
    - Vorstellen mit Füßen wegzuziehen

12. Train Pull
    -Vorstellen mit Armen herzuziehen

13. Aktionen durchführen (Vordruck benutzen)

14. Zwischenfragebogen Explicit Interaktion

15. Abschlussfragebogen

Aktionen für explizite Interaktion

Teilnehmernummer: _____

Aktion	Erkannt	Nicht erkannt
Pull		
Push		
Push		
Pull		
Pull		
Push		
Pull		
Push		

MediaBrain: a Video Annotation tool based on Brain-Computer Interaction

Sahami Shirazi Alireza, Funk Markus, Glück Hendrik, Pfleiderer Florian, Schmidt Albrecht

Chair of Human-Computer-Interaction, VISUS, University of Stuttgart

Abstract

Adding notes to time segments on a video timeline makes it easier to search, find, and playback important segments of the video. Various approaches have been explored to annotate videos (semi) automatically in order to summarize videos. In this research we investigate the feasibility of annotating videos based implicitly on brain signals retrieved from a Brain-Computer Interface (BCI) headset. The signals provided by the used BCI can reveal different information such as brain activities, facial expressions, or the level of excitement about users. This information correlates with scenes the users watch in a video. Thus, it can be used for annotating a video and automatically generating a video summary. In order to achieve the goal, an annotation tool called MediaBrain is developed and a user study conducted. The result reveals that it is possible to annotate a video and generate a set of highlights based on the excitement information.

1 Introduction

In this research we investigate the feasibility of annotating videos implicitly based on the information provided by the BCI (Brain Computer Interface). Adding notes to time segments on a video timeline makes it easier to search, find, and playback important segments of the

video. Various approaches have been explored to annotate videos (semi) automatically in order to summarize videos. Annotations are either defined automatically by analyzing and processing videos or explicitly by users/annotators.

The brain is the center of the nervous system. The BCI enables communication between a brain and an external device. Several methods such as Magnetoencephalography (MEG), functional Magnetic Resonance Imaging (fMRI), and electroencephalograms (EEG) can acquire the brain signals. EEG is one of the most popular methods due to the ease of use. It uses non-invasive number of electrodes spread over the scalp to measure signals arising from neural activities. This allows to measure and analysis the brain neural activity without complex medical procedures. The brain neural activity varies based on the mental and cognitive activities.

The user interaction with the computer is categorized into explicit and implicit interactions. In the explicit interaction the user aims for a certain action and expects certain results. While in the implicit interaction the user is not primarily aimed to interact with the computer, but the computer understands as an input (Schmidt, 2000). Researchers have explored various resources such as the eye movement (Santella *et al.*, 2006, Buscher *et al.*, 2008) for implicit interactions. In this paper we explore implicit video annotating based on the information provided by the BCI.

The BCI has been primarily used in the medical and clinical fields. However, with advantages in technologies low-cost commercial wireless EGG headsets are available for other purposes, e.g., game (Myeung-Sook *et al.*, 2010). In this research we utilize brain signals as an implicit input for annotating time segments videos and generating a set of highlights. Brain signals can reveal different information such as facial expressions or the level of excitement about users. This information correlates with scenes the users watch in a video. Thus, it can be used for annotating a video and generating a summary. To achieve the goal, we develop an annotation tool and conducted a user study. Based on the authors' knowledge, this is the first step toward using brain signals for annotating videos.

2 Related Work

There are various methods to measure brain signals. The MEG (Magnetoencephalography) technique maps the brain activity by recording magnetic fields produced by electrical currents occurring naturally in the brain. fMRI (functional Magnetic Resonance Imaging) measures brain activity by detecting associated changes in blood flow. EEG (Electroencephalograms) measures brain voltage fluctuations resulting from ionic current flows within the neurons. It measures six signals each with different frequency ranges (alpha: 8–13 Hz, beta: 13–30 Hz, gamma: 30–100+ Hz, delta: up to 4 Hz, theta: 4–8 Hz, mu: 8–13 Hz). With the advantage in new technologies, commercial EEG headsets are recently available and can be used for laboratory studies. The two most popular ones are NeuroSky[1]

[1] http://neurosky.com/ (accessed March 2012)

and Emotive EPOC[2] headsets. The NeuroSky devices have two electrodes and distinguish neutral and attentive mental states with 86% accuracy (NeuroSky, 2009). The Emotiv EPOC headset has 14 data collecting electrodes and 2 reference electrodes, detects various facial expressions, level of engagement, frustration, mediation, and excitement.

Both headsets have been used in various research projects. *ThinkContacts* is an application that uses the NeuroSky MindSet to measure the degree of attention each contact gets in an address book to find out which contact to call (Perkusich *et al.*, 2011). *Neurowander* is a BCI game using brainwaves as inputs for a game (Myeung-Sook *et al.*, 2010). Mostow *et al.*, 2011, used the EEG headset to collect and assess cognitive information from students while reading different texts. The suitability of the NeuroSky MindSet to measure and categories a user's level of attention and mediation was assessed in Crowley *et al*., 2010. Furthermore, Petersen *et al.,* 2011, demonstrated the ability to distinguish emotional responses reflected in different scalp when viewing pleasant and unpleasant pictures. Yasiu, 2009, analyzed the mental state of a car driver using a mobile phone while driving and proposed a data processing technique for daily life applications. Lotte *et al.*, 2007, provided an overview on classification algorithms for EEG-based brain-computer interfaces.

Other researches used the ERP (Event Related Potential) wave for interaction with a system. The ERP wave is the brain response that is directly the result of a thought or perception. The P300 wave is the famous ERP elicited in the process of decision-making. Kanoh *et al*. used the P300 signal for controlling the mouse course. It works by cycling through 8 possible directions around the current cursor position. When the signal is triggered, the mouse moves into the desired direction (Kanoh *et al*., 2011). Li *et al*. developed a P300-based keyboard that basically works by cycling through all letters until the desired one is reached (Li *et al.*, 2009). *NeuroPhone* is a system that uses ERP signals obtained from the EPOC headset to select a contact from an address book on an iPhone and dial the number (Campbell *et al.*, 2010). One of the brain signal monitoring usages in the medical domain is sleep monitoring in order to help people with sleeping disorders. Although the brain signals during sleep are quite weak and difficult to read (Garg *et al*., 2011).

Beside, researchers have investigated various automatic or semiautomatic approaches for annotating videos. Yamamoto *et al*., 2008, used the social activity, i.e., users' comments and weblog authoring for annotating videos. Nagao *et al.,* 2002, provided an annotation tool allowed users to easily create annotations including voice transcripts, video scene descriptions, and visual/auditory object descriptions. Sahami Shirazi *et al.*, 2011, used an iconic interface on the mobile phone for sharing opinion during sport events, annotating the events, and detecting highlights. Nakamura *et al.*, 2008, explored affective response to understand video commenting systems. Various algorithms also tried to automatically annotate videos (Wang *et al.*, 2009, Lavrenko *et al.*, 2004). Saur *et al.* developed a tool, which automatically annotated basketball videos based on video content (Saur *et al.*, 1997).

[2] http://www.emotiv.com/ (accessed March 2012)

None of the previous work utilized emotional information for annotating a video. Emotional reactions of a video viewer to scenes in a video are correlated to what happens in a scene. In our research we used the brain signals acquired from the Emotiv EPOC headset to annotate a video and find highlights. In order to achieve this, we developed a prototype described in the next section.

3 MediaBrain

To annotate videos with emotional information acquired from the brain, the Emotiv EPOC headset was used and an annotation tool, called MediaBrain, was developed.

3.1 Annotation tool

We used the Emotiv EPOC headset to obtain the brain signals during watching a video. The headset uses EEG method and has 14 electrodes and 2 reference electrodes. It transmits data to a Windows based computer via a Bluetooth connection. The SDK (Software Development Kit) that comes with the headset provides following measurements: facial expressions, level of engagement, frustration, mediation, and excitement. The headset has also a built in gyroscope that detects the user's head orientation. The sampling rate is 128 samples/second.

A video annotation tool called MediaBrain is developed as an application for a personal computer. The tool includes the open source VLC[3] video player for fully controlling the video events such as playback, pause, or stop a movie. It is implemented in Visual C++. MediaBrain establishes a Bluetooth connection with the EEG headset and records the user's brain signals while playing a video. It uses the EPOC's SDK to retrieve the brain information. The brain information is recorded and stored in an XML file tagged with the video timestamp. In the current version just excitement values are recorded and used to annotate a video. However, it is easily possible to extend the tool and use other parameters for the annotation. After gathering the information, the tool uses the XML file to identify, extract, and play scenes highlighted based on the excitement information.

[3] http://www.videolan.org/vlc/ (accessed March 2012)

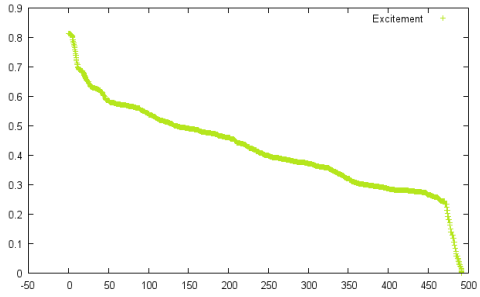

Figure 1:Sorted user excitement values

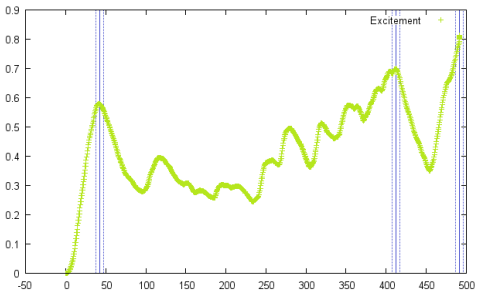

Figure 2:User excitement graph with calculated highlights

3.2 Annotation Algorithm

An algorithm is developed to extract the highlights based on the recorded information. The algorithm requires two parameters: length of a highlight (*L*) in seconds and maximum number of highlights (*N*). In the first step all highlights are sorted in descending order based on the excitement value (see Figure 1). Then, a highlight segment is detected. To do so, the scene that has the maximum excitement value together with the ±*L/2* seconds is extracted. The other excitement values in this time segment are excluded for further calculation. This procedure is continued till the maximum number of highlights (*N*) is calculated or no more data is available. If *N* is not provided, all available points are extracted. Figure 2 depicts the excitement graph with calculated highlights. The algorithm is described in Table 1.

Table 1: The pseudo code describes the algorithm for annotating the scenes with the excitement values.

```
01 N = maximum number of highlights;
02 L = length of a highlight;
03 excitement_array = sort the excitement data in a descending
                      order;
04 For 1 till N:
05     item = Select first item in excitement_array;
06     highlight_start_time = item.timestamp - L/2;
07     highlight_end_time   = item.timestamp + L/2;
08     in excitement_array remove data from highlight_start_time
       till highlight_end_time;
```

4 User Study

A user study was conducted in order to evaluate the MediaBrain tool and assess the feasibility of annotating the video based on the excitement information provided by the Emotiv EPOC headset.

4.1 Apparatus

We had 11 participants (7 male, 4 female, average age 23.2) for the user study. All participants were students recruited via mailing lists and university's forums. In the first step, each participant was asked to answer a questionnaire about the demographics. Then, we continued with watching a video. The participant wore the headset and started watching a short animation movie, called Big Buck Bunny[4]. We assured that all the 16 electrodes had a good connection to the scalp. We selected this movie as it had few funny scenes that should result in a distinct excitement graph. The movie's length was 10 minutes and shown on a 40" display. Figure 3 shows the setup of the user study. The participant wore the EPOC headset while watching the animated movie on the big screen. The experimenter was able to check the live data on the second monitor.

Along with the excitement data obtained implicitly from the EPOC headset, the users were asked to explicitly specify their excitement while watching the movie. Hence, we extended the MediaBrain tool in a way that users could state their excitement by pressing a button. During the study we asked the participants to press the button every time they were excited about a scene. This information was stored together with the video timestamp in the XML file and used later for the evaluation. At the end of the study, the participants filled in another questionnaire and provided qualitative feedback about their experience during the study. The study took approximately 30 minutes for each participant.

```
01 N = maximum number of highlights;
02 L = length of a highlight;
03 excitement_array = detect points where the gradient changes;
04 For 1 till N:
05     item = Select first item in excitement_array;
06     highlight_start_time = item.timestamp - L/2;
07     highlight_end_time   = item.timestamp + L/2;
08     in excitement_array remove data from highlight_start_time
       till highlight_end_time;
```

Table 2: The pseudo code describes the updated algorithm for annotating the scenes with the excitement values.

4.2 Result

Based on the demographic questionnaire, 70% of the participants daily used their computer for watching videos. None of the participants took part in any user study related to the BCI or used any type of BCI headsets. One user saw the animation shown in the study before.

The results revealed that the participants pressed the button (specified their excitement explicitly) 13 times on average. There were 6 scenes where 85% of the users pressed the

[4] www.bigbuckbunny.org (accessed March 2012)

button. The rest of the explicit highlights were widely spread. The 6 scenes included mainly unexpected actions in the movie, led to a surprise and excitement.

We also investigated the correlation between the explicit and implicit excitement information from users. We took each explicit input from users and checked whether this input matched with a highlight detected by the algorithm. The results showed that with L=5 seconds only 27% of explicit inputs matched with the implicit excitements. With L=10 seconds the result was 36%. Further investigation revealed that the user inputs were on average 10 seconds earlier than the local maximum excitement values. The user inputs matched with the points where the excitement level started increasing (changes in gradient). However, we expected that the explicit inputs located on the local maximums (peaks) in the excitement graph (see Figure 4). Based on the Model Human Processor (Card, *et al.*, 1986) the total cycle time of processors in the human's cognitive system, namely the *perceptual*, the *cognitive*, and the *motor processor* is approximately 300 msec. Therefore, we updated our algorithm in a way that the points where the excitement level started increasing were considered as highlights (see Table 2). Based on the updated algorithm we analyzed the data again. The results showed that with the new algorithm 65% of the users inputs overlapped with the highlights extracted via the algorithm.

The qualitative feedback showed that all users were relaxed during the study and enjoyed watching the movie. None of the users found the interaction with the EPOC headset inconvenient or disturbing. Also, all users mentioned that the explicit defining of excitements was not distracting them from concentrating on the movie. 77% of the users stated that they could imagine using the system in daily situations, like in front of the TV.

Figure 3: User taking part in the study

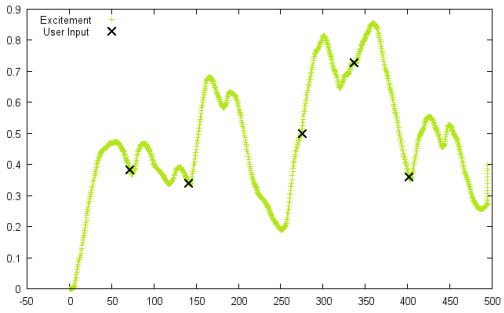

Figure 4:User-Highlights mapped to the excitement graph

5 Discussion & Conclusion

In this research we aimed at annotating videos based on excitement information acquired from a BCI headset. Hence, an annotation tool was developed and a user study was conducted.

The results reveal correlations between the scenes in the movie and the excitement level. Videos can be annotated with excitement information obtained from the EPOC headset and highlights can be extracted. Though the local maximums in the excitement graph correlates with the highlights in the video, but these are not the moments users believe they are excited. Moments which users think they get excited are the points where the excitement value starts increasing (gradient changes) in the excitement graph. Therefore, it is important to consider points where the excitement level start increasing instead of the peaks in the excitement graph for annotating a video or generating a summary.

This research shows it is feasible to implicitly annotate a video based on excitement information and generate a set of highlights. Users emotional reactions during watching a video are rich resources for implicitly annotating and extracting important scenes in a video. In this research only excitement information was explored.

We are currently planning to investigate whether other emotional information such as frustration or facial expressions can be used for annotating a video. On the other hand, sharing the emotional reactions during watching a movie between non-collocated viewers might result in an increase in the connectedness and awareness among them.

Reference

Buscher, G., Dengel, A., & van Elst, L. (2008), *Query expansion using gaze-based feedback on the subdocument level*. In Proceedings of ACM SIGIR'08 conference on Research and development in information retrieval , ACM (2008), S. 387–394.

Campbell, A., Choudhury, T., Hu, S., Lu, H., Mukerjee, M.K., Rabbi, M. & Raizada, R.D.S. (2010). *Neurophone: brain-mobile phone interface using a wireless eeg headset*. In Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds, S. 3-8

Card, S., Moran, T., and Newell, A. The model human processor. In Kenneth R. Bo, Lloyd Kaufman, and James P. Thomas, editors, Handbook of Perception and Human Performance, chapter 45. John Wiley and Sons, 1986.

Crowley, K., Sliney, A., Pitt, I & Murphy, D. (2010). *Evaluating a brain-computer interface to categorise human emotional response.* In Advanced Learning Technologies (ICALT), 2010 IEEE 10th International Conference, 276-278

Garg, G., Singh, V., Gupta, J.R.P., Mittal A.P. & Chandra, S. (2011). *Computer assisted automatic sleep scoring system using relative wavelet energy based neuro fuzzy model*. In WSEAS Transactions on Biology and Biomedicine, 8

Kanoh, S., Miyamoto, K. & Yoshinobu, T. (2011). A p300-based bci system for controlling computer cursor movement. In Conference proceedings: Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, volume 2011, S. 6405

Lavrenko, V., Feng, S.L. & Manmatha, R. (2004), *Statistical Models for automatic video annotation and retrieval*. In Proceedings. (ICASSP '04). IEEE International Conference on Acoustics, Speech, and Signal Processing

Li, Y., Zhang, J., Su, I. Chen, W., Qi, Y., Zhang, J. & Zheng, X. (2009). *P300 Based BCI messenger*. In Complex Medical Engineering, 2009. CME. ICME International Conference, S. 1-5

Lotte F., Congedo M., Lecuyer A., Lamarche F. & Arnaldi B. (2007), *A review of classification algorithms for EEG-based brain–computer interfaces.* In Journal of Neural Engineering, Volume 4, Number 2

J. Mostow, K. Chang, & J. Nelson (2011). *Toward exploiting eeg input in a reading tutor.* In Artificial Intelligence in Education, S. 230-237

Myeung-Sook, Y., Joonho, K., Sunghoon, K. (2010). *Neurowander: a bci game in the form of interactive fairy tale.* In In Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing, Ubicomp '10 Adjunct, S. 389-390

Nagao, K., Ohira, S. & Yoneoka, M (2002), Annotation-based multimedia summarization and translation. In Proceedings of the 19th international conference on Computational linguistics - Volume 1 (COLING '02), S. 1-7

Nakamura, S., Shimizu, M. &Tanaka, K (2008). *Can social annotation support users in evaluating the trustworthiness of video clips?* In Proceedings of the second Workshop on Information credibility on the web, S. 59–62.

NeuroSky (2009), *NeuroSky's eSense™ meters and Detection of Mental State*. Neurosky, Inc.

Perkusich, M.B., Rached, T.S. & Perkusich, A. (2011). *Thinkcontacts: Use your mind to dial your phone.* In Consumer Electronics (ICCE), 2011 IEEE International Conference, S. 105-106

Petersen, M.K., Stahlhut, C., Stopczynski, A., Larsen, J.E. & Hansen, L.K. (2011*), Smartphones Get Emotional: Mind Reading Images and Reconstructing the Neural Sources.* In Proceedings of fourth International Conference on Affective Computing and Intelligent Interaction

Rebolledo-Mendez, G., Dunwell, I., Martínez-Míron, E. Vargas-Cerdán, M., De Freitas, S., Liarokapis, F. & García-Gaona, A. (2009). *Assessing Neuroskys usability to detect attention levels in an assessment exercise.* In Human-Computer Interaction, New Trends, 149-158

Sahami Shirazi, A., Rohs, M., Schleicher, R., Kratz, S., Müller, A. & Schmidt, A. (2011), *Real-time nonverbal opinion sharing through mobile phones during sports events*. In Proceedings of ACM Conference on Human Factors in Computing Systems, New York, NY, USA, S. 307-310.

Santella, A., Agrawala, M., DeCarlo, D., Salesin, D., & Cohen, M. (2006), *Gaze-based interaction for semi-automatic photo cropping.* In Proceedings of ACM Conference on Human Factors in Computing Systems '06 , ACM (New York, NY, USA, 2006), S. 771–780.

Saur, D.D., Tan,Y.-P. Kulkarni, S.R. & Ramadge, P. J. (1997), *Automated analysis and annotation of basketball video*. In Proceedings of SPIE's Electronic Imaging conference on Storage and Retrieval for Image and Video Databases V, S. 176-187

Schmidt, A. (2000), *Implicit human computer interaction through context*. In Personal and Ubiquitous Computing, S. 191-199

Wang, M., Hua, X.-S., Hong, R., Tang, J., Qi G.-J, Yan Song (2009), *Unified Video Annotation via Multigraph Learning.* In IEEE Transactions on Circuits and Systems for Video Technology, Volume 19, Issue 5, S.733 - 746

Yamamoto, D., Masuda, T., Ohira, S. & Nagao, K. (2008), *Video Scene Annotation Based on Web Social Activities*. In IEEE MultiMedia Volume 15, Number 3: S. 22-32

Yasui, Y. (2009), *A brainwave signal measurement and data processing technique for daily life applications.* In Journal of physiological anthropology, Volume 28, Number 3: S. 145-150

Erklärung

Hiermit versichern wir, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift: _____

Markus Funk, Stuttgart den 11.05.12

Unterschrift: _____

Hendrik Glück, Stuttgart den 11.05.12

Unterschrift: _____

Florian Pfleiderer, Stuttgart den 11.05.12