Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit

# Extracting Facial Data using Feature-based Image Processing and Correlating it with Alternative Biosensors Metrics

Clint Banzhaf

| | |
|---|---|
| **Course of Study:** | Softwaretechnik |
| **Examiner:** | Prof. Dr. Albrecht Schmidt |
| **Supervisor:** | Thomas Kosch, M.Sc.<br>Mariam Hassib, M.Sc.<br>Dipl.-Inf. Benjamin Jillich |
| **Commenced:** | 25. Juli 2016 |
| **Completed:** | 24. Januar 2017 |
| **CR-Classification:** | I.2.10<br>I.4.8<br>I.5.4 |

# Abstract

Extracting facial data has recently drawn more attention due to an ongoing progress in research and increase in its accuracy. Thanks to mobile platforms, the necessary but ordinary cameras are remarkably widespread nowadays. Big companies, like Google, have entered the market with their own cloud-based and mobile face analysis solutions. Possible applications for extracted facial data range from more user-friendly devices to user-interface evaluation and marketing. This thesis starts with an introduction to the topic of facial data extraction. It outlines several interesting applications, important techniques and existing frameworks first. Afterwards, the main contribution is presented: *OpenFace++*, an improved and extended variant of the existing *OpenFace* face analysis framework. Like *OpenFace*, the proposed *OpenFace++* framework aims to close an unsatisfying gap in available functionality between commercial and free open-source solutions. *OpenFace++* adds several new features to *OpenFace*. Amongst others, the added highlights are facial expression recognition, attention estimation, and cross-platform compatibility. *OpenFace++* works on Android out-of-the-box. The work is completed by presenting a user-study which was conducted to investigate the reliability of extracted facial data. The first experiment evaluates the detectability of eye-closeness states. Results show that facial data extraction can outperform competing solutions. The second experiment shows that face orientation extraction also works reliably compared to a gyrometer. Finally, the third experiment demonstrates that the new facial expression detection can detect many cases very accurately. The thesis finishes with some suggestions for further improvements on *OpenFace++* in the future.

# Kurzfassung

Das Extrahieren von Gesichtsinformationen hat in letzter Zeit wegen anhaltenden Fortschritten in der Forschung und einem Anstieg in seiner Genauigkeit erhöhte Aufmerksamkeit auf sich gezogen. Dank der mobilen Plattformen sind die dafür notwendigen aber gewöhnlichen Kameras heutzutage bemerkenswert weit verbreitet. Große Unternehmen wie Google sind mit ihren eigenen cloud-basierten und mobilen Gesichtsanalyse-Lösungen in den Markt vorgestoßen. Denkbare Anwendungen für extrahierte Gesichtsinformationen reichen von nutzerfreundlicheren Geräten über Evaluierung von Benutzerschnittstellen bis hin zu Marketingaspekten. Diese Diplomarbeit beginnt mit einer Einführung in das Thema Gesichtsinformationsextrahierung. Sie skizziert zuerst einige interessante Anwendungsmöglichkeiten, sowie einige wichtige Techniken und bestehende Frameworks. Anschließend wird der Hauptbeitrag vorgestellt: *OpenFace++*, eine verbesserte und erweiterte Variante des bestehenden *OpenFace* Gesichtsanalyse-Frameworks. *OpenFace++* versucht, wie auch *OpenFace*, eine unbefriedigende Lücke in der verfügbaren Funktionalität zwischen kommerziellen und kostenlosen open-source Lösungen zu schließen. *OpenFace++* erweitert *OpenFace* um einige neue Funktionen. Die hinzugefügten Highlights sind unter anderem eine Erkennenung für Gesichtsausdrücke, eine Abschätzung der Aufmerksamkeit und plattformübergreifende Kompatibilität. *OpenFace++* läuft auf Android out-of-the-box. Die Arbeit wird abgeschlossen durch die Vorstellung einer Nutzerstudie, welche durchgeführt wurde, um die Zuverlässigkeit der extrahierten Gesichtsinformationen zu untersuchen. Das erste Experiment evaluiert die Erkennbarkeit des Geschlossenheitszustandes der Augen. Die Ergebnisse zeigen, dass das Extrahieren von Gesichtsinformationen konkurrierende Verfahren übertreffen kann. Das zweite Experiment zeigt, dass das Extrahieren der Gesichtsausrichtung ebenfalls zuverlässig funktioniert, verglichen mit einem Gyrometer. Das dritte Experiment demonstriert zuletzt, dass auch die neue Gesichtsausdruckserkennung in viele Fällen akkurat funktioniert. Die Diplomarbeit endet mit einigen Ratschlägen für weitere Verbesserung an *OpenFace++* in der Zukunft.

# Contents

# List of Figures

# List of Tables

# 1 Introduction and Motivation

Human perception can extract significant amounts of information from only a quick glance at another person's face. We almost instantly detect human faces in images and it takes us less than a second to classify our counterpart's facial expression into categories like happy (friendly) or angry (dangerous). These skills originate from evolutionary pressure to possibly take necessary actions immediately, they are indispensable in our everyday social behavior and usually still work fine even in bad conditions, such as occlusions, low illumination, or different poses.

However, what seems like an easy task for humans, still yields a notable challenge for computers and other kinds of artificial intelligences. Hence, improving facial data extraction has been a focus of computer scientists for a long time. By now, several popular approaches, especially for common intermediate steps, such as face detection, have been presented. Additionally, work exists in the field of facial expression detection, gender and age estimation or head pose and motion detection. Still, none fully reaches the accuracy of human perception yet and many can be outsmarted on purpose.

The great advantage of collecting information via extraction from images or videos is the fact that it is completely non-intrusive for the user. He neither has to wear special equipment nor has to stay in close physical contact with any other kind of hardware. The user also does not have to pay any attention to specifics on how to operate the detection system. Furthermore, cameras are small, have become a cheap bulk good and are available practically everywhere, especially with regard to the popularity of mobile platforms. This puts facial data extraction by ordinary webcams at a great advantage compared to other highly specialized sensors.

There are many promising ideas for applications which could greatly benefit from extracted facial data: Imagine a car which observes the driver through a camera and recognizes when he falls asleep or does not pay enough attention to the street. Or think about mobile apps that are aware of the user's mood by observing him while he is using the user interface. There are also applications in the field of marketing and merchandising that can make great use of extracted facial data to gain feedback from customers.

Next, some ideas are described in more detail.

1. **Driver Assistance System**

   Several car manufacturers have already begun to develop smarter cars. However, this ideally requires collecting information about the current physical and mental state of the driver.

   Using this information, the car could be able to improve its behavior with regard to safety. For instance, it could play a warning tone if the driver keeps his eyes closed for more than a second. Such a system may also be able to detect the current orientation of the driver's face and recognize when he is not paying enough attention to the street and traffic in front of the car, because he is looking out of the side window at that moment. Furthermore, the car may be able to detect that the driver is in an angry mood and is driving way too fast. As a consequence it may warn the driver or take other actions to defuse the situation.

   A camera integrated into the armature or some other place would observe the driver and forward captured frames to the processing system which then may take appropriate actions to assist the driver.

**Figure 1.1:** Driver Assistance System[1]

[1] Volvo Vision 2020 Project
http://blog.caranddriver.com/volvo-tests-driver-alertness-with-face-monitoring-technology-yeah-its-creepy/

2. **Marketing and Merchandising**

   Collecting meaningful customer statistics or marketing feedback is a non trivial task. Handcrafting manual studies with tailored questionnaires or observing customers by human agents is an expensive, yet often still inevitable evil.

   Facial data extraction could be a great step towards tackling these problems. Automated systems can keep track of customers entering the shop and analyze their marketing relevant behavior. For instance, which products draw the most attention and what positive or negative emotions the proposed offers cause. Additionally, information about gender or age can help to classify the customer into groups and provide optimized advertisement or special product offers.

   Not surprisingly, several commercial systems related to these tasks have been presented already. The extracted information ranges from simple customer counting, up to advanced data extraction including gender, age, mood and attention time estimation.

   Figure 1.2 shows a presentation slide from the company Sightcorp, one of many which offer commercial solutions based on extracted facial data.

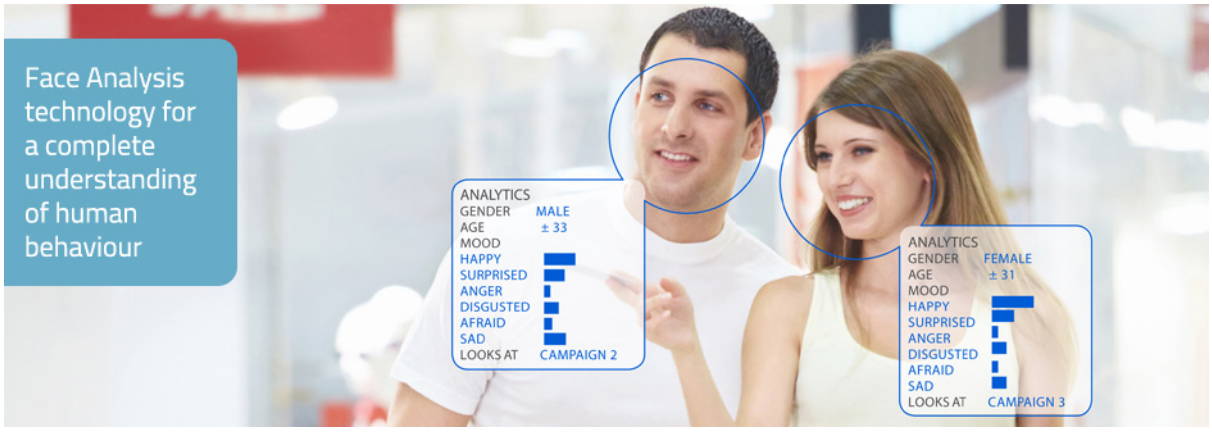


**Figure 1.2:** Commercial Face Analysis from Sightcorp[1]

[1] Image by Sightcorp (http://sightcorp.com)

3. **User Interface Evaluation and Improvement**

   Extracted facial data can contribute greatly to the evaluation of user interfaces and it can also assist in creating more user aware and adaptive interfaces.

   For instance, observing a user's facial expressions can help a lot in understanding his or her perception of the presented input masks. Users who tend to show confused, annoyed or angry facial expressions are possibly not understanding the presented user interface correctly and might need further assistance.

   Additionally, extracting the user's viewing direction (eye-gaze) and face orientation can help in estimating the attention that is currently paid to the application or its user-interface. For instance, the display brightness can be dimmed in case the user is facing into a different direction. The extracted eye-gaze can also be used to calculate a focused point on the display. The accuracy however is limited and not as good as with professional infrared based installations.

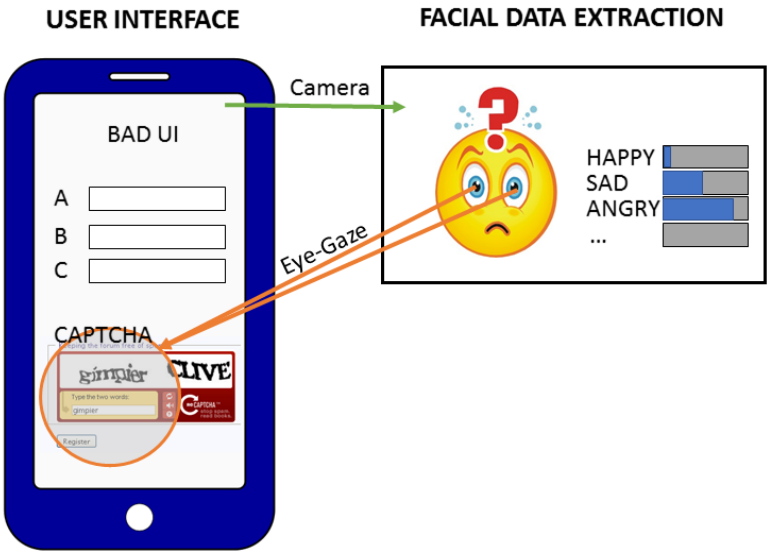


**Figure 1.3:** User Interface Evaluation

4. **Guided Mental Training**

   Many exercises have been presented to improve or preserve our mental skills, to calm down our mind or to learn shifting and maintaining our focus on command. However, the proper application of these exercises is very important and therefore typically requires guided and supervised training.

   Furthermore, these exercises can require expensive biosensor hardware, for instance an EEG or heart-rate monitor, and therefore come with a great initial hurdle for beginners.

   The capabilities of extracted facial data are limited for this purpose, still they are sufficient for basic supervision. For instance, it's typically indispensable for meditation exercises to keep head motions to a minimum and to keep the eyes closed. These requirements can be validated using extracted facial data.

   Imagine the user in front of and supervised by an application running on his or her own smart phone.

**Figure 1.4:** Guided Mental Training

## Outline

The further chapters are structured as follows:

**Chapter 2 – Background and Related Work:** In this chapter, the reader is introduced to necessary background information. A short introduction to machine learning classification is given. Different techniques and tools for extracting facial data from ordinary webcams are presented. Also an introduction to other biosensors, such as an *Electroencephalogram* (EEG), which at least partially cover the same functionality, is included here. Finally, some existing and later used computer vision and machine learning frameworks are described in this chapter.

**Chapter 3 – OpenFace++:** This chapter describes the *OpenFace++* library. The library has been created for this thesis. Included are further details about its implementation, the improvements over the original *OpenFace* library, its Android support and how to use the library correctly. Additionally, the chapter introduces the *Face Sensor Application*, an application used in the study and a demonstration of the features of the *OpenFace++* library.

**Chapter 4 – User Study:** In this chapter, the user study which was conducted as part of the thesis is presented. The study aims to validate the accuracy of the presented *OpenFace++* library and its facial data extraction. Furthermore, the study compares the results against other biosensors and explores the chances to either replace the sensors or improve the overall accuracy by combining them.

**Chapter 5 – Conclusion and Future Work:** This chapter presents the summary and conclusions. Knowledge gained from the user-study is outlined and possible future improvements and extensions for the presented *OpenFace++* library are proposed.

# 2 Background and Related Work

## 2.1 Machine Learning Basics

This section gives a brief introduction to the field of machine learning. Since it's a big research field, only those basics are addressed which are required to understand the mechanics behind facial data extraction.

### 2.1.1 Classification and Regression

*Classification* and *regression* are both mathematical functions and as such require an input and return an according output. In both cases, the input is typically described by many parameters at once (high dimensional vector), while the output is usually only a single value (scalar).

For *classification*, the output is from a *finite* and *discrete* range, such as any range within the *natural numbers* ($\mathbb{N}$). A popular case is a *binary classifier*, which only returns *true* or *false*. However, classifiers returning a color from the set {*Red*, *Blue*, *Black*} or an integer from the interval [1..5] are also valid examples.

For *regression*, the output is from a *finite* and *continuous* range, such as any arbitrary range within the *rational numbers* ($\mathbb{Q}$). The range is only roughly defined by the used maximum and minimum values in the positive and negative samples, but is not a strict boundary on the output value. A popular choice for the output range is the interval between 0.0 and 1.0.

The disadvantage of *classification* can be its limited fineness on the output compared to *regression*. Let's consider the following small example:

You want to create a machine learning classification or regression to know whether a customer is likely interested in your product or not. Let our customer be described by five attributes and hence be a point in five dimensional space. For instance, let the customer's attributes be: *age*, *gender*, *ethnicity*, *mood* and *has-glasses*. Of course it's crucial for success in real cases to find the right attributes. Those that influence the actual decision most. Now you must go capture *positive* and *negative* samples. Collect

the mentioned five attributes of as many customers as possible. Of those who bought the product (*positives*) and of those who were interested, but then skipped it (*negatives*). Then, without running any complicated statistics or investigating the data yourself, you can train your favorite machine learning classification or regression on the samples to receive a prediction for any unseen customer. To do so, the *positives* are assigned a value of 1 in classification and a value of 1.0 in regression. The *negatives* are set to 0 in classification and to 0.0 in regression. Afterwards the training process is started and it tries to find a good separation between the positives and negatives. The details very much depend on the actual technique.

For a *classification* approach, the output in our example will be a *true* (buy) or *false* (skip) prediction. For *regression* it will be between 0.0 and 1.0 and can be seen as the likelihood at which the unseen customer is going to buy our product.

There might be some easily visible or known correlation between some attributes and the final decision. For instance, a product designed for women is already known to have a higher chance to be bought by female customers. However, the machine learning approaches will easily recognize these correlations. Furthermore, they use sophisticated techniques to discover and make use of unknown and not easily recognizable correlations.

### 2.1.2 Support Vector Machines

*Support Vector Machines* (SVM) are one of many machine learning approaches.

This approach became popular after being presented by Cortes and Vapnik [SVM95], though some of the mathematical foundations are much older.

*Support Vector Machines* can do classification (SVC) as well as regression (SVR). Their training process involves finding the high dimensional hyperplane, which separates the positive and negative samples best.

The hyperplane does not necessarily need to be linear as shown in Figure 2.1 in red. Polynomial, radial and other variants are possible. Once trained a SVM can predict an output for any unseen input sample using the found separation.

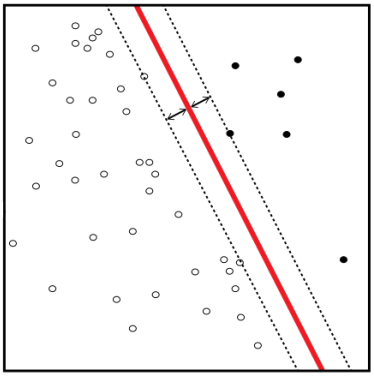

**Figure 2.1:** Hyperplane

## 2.2 Extracting Facial Data

In this section, state of the art techniques and tools for extracting facial data are presented. Image analysis with regard to faces has been a focus of computer scientists for a long time. One of the most referenced early pioneers was presented by Kanade [Kanade73]: A hardware driven face image processing system built from scratch. However, due to the lack of computing power and other limitations, it took until the 90s to bring up reasonably working solutions in the field of face detection, recognition and further analysis. The possible applications for extracted facial data are manifold and range from security related interests to usage in medical context or even advertisement.

### 2.2.1 Face Detection

Face detection refers to the task of finding human faces in arbitrary input images. It is often referred to as *in-the-wild* detection. The faces may appear at any scale, position and orientation. They can also be affected by occlusions or heavy differences in lighting conditions. Additionally, the input image might not contain any faces at all.

Face detection is typically the first and often independent step in a larger approach to extract facial data from *in-the-wild* environments. The output is a subsection of the original image, supposed to contain a face and usually described by a simple shape, such as a rectangle or ellipse. This bounding shape then serves as input for further analysis. Figure 2.2 shows the output of two popular detectors applied on the same video frame.

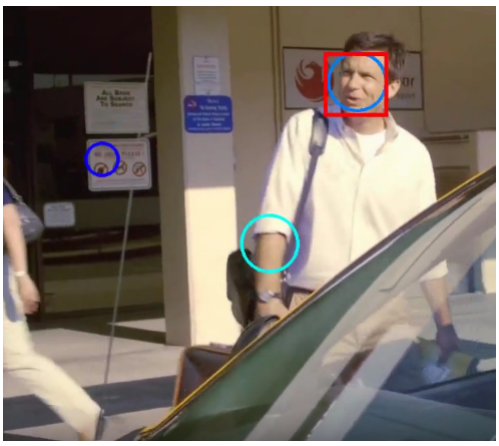

**Figure 2.2:** Detections by [Viola-Jones01] (circles) vs. [HOG-SVM05] (rectangles)

Several different systems have been proposed for face detection over the last twenty years. They cover many of the available machine learning approaches. In the nineties, Rowley, Baluja, and Kanade [Rowley96] presented a artificial neuronal network based detector and Osuna, Freund, and Girosi [Osuna97] used Support Vector Machines to address the face detection problem.

Viola and Jones [ViolaJones01] proposed a boosted cascade of weak classifiers with simple rectangular features and was considered a major breakthrough at that time. Each weak classifier in the cascade checks for a selected rectangular feature as shown in Figure 2.3. The features are cheap to calculate using an integral image and are selected by an AdaBoost learning algorithm. They refer to properties of the face, such as the area around the eyes being typically darker than the area around the cheek below. Each weak classifier in the cascade must have a very low false-negative rate, because a once made reject decision can not be corrected anymore. But on the other hand, each classifier in the cascade only requires a low true-positive rate, because there are plenty of more classifiers which can still reject the sample. This kind of detector outperformed its earlier competitors clearly with regard to detection rate, false-positive robustness and computational costs. It also became popular due to its implementation being available as part of the free *OpenCV* (see 2.4.1.1) framework.

**Figure 2.3:** [ViolaJones01] Rect.-Features

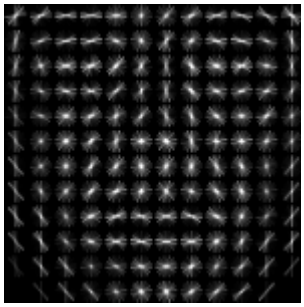

**Figure 2.4:** [HOG-SVM05] HOG-Features

The next major step was presented by Dalal and Triggs [HOG-SVM05]. The authors introduced a new type of feature called *histogram of gradients* (HOG) and combined it with *Support Vector Machines* (SVM). The HOG features are calculated on many rectangular sub-regions of the input image. Figure 2.4 shows a HOG feature visualization of a human face. Afterwards, *Support Vector Classification* (SVC) is applied on the extracted feature values to determine whether the current candidate represents a face or not. This approach has proven to show a lower false-positive rate at higher computation cost than the solution presented by Viola and Jones [ViolaJones01] and it comes with a freely available implementation as part of the *dlib* (see 2.4.1.2) framework.

### 2.2.2 Face Recognition

*Face Recognition* refers to the task of matching an unseen face image against an existing database. The input image is typically strongly constrained and usually the output of an earlier preprocessing step: It has proven to contain a human face and the face is perfectly aligned and cropped to match the examples from the database.

Turk and Pentland [Eigenfaces91] used *Principle Component Analysis* (PCA) to create *Eigenfaces* from a set of training images. Projecting an unseen sample into the learned PCA space then allows assigning the label. Belhumeur, Hespanha, and Kriegman [Fisherfaces97] replaced the PCA with an improved *Fisher's Linear Discriminant* (FLD) and showed its improvements over the original approach. Ahonen, Hadid, and Pietikäinen [Ahonen04] later used *Local Binary Pattern* (LBP) to tackle the problem.

In theory, these approaches can be used for any kind of facial *compare-and-label-it* task, including emotion estimation. After capturing the facial expressions from a subject, any unseen sample from him or her could be matched against these. But in practice, the approaches are not robust against changes in lighting or pose. Small differences in the environment cause these approaches to fail and training a per subject dataset is often not acceptable. Hence more advanced and model based approaches were presented.

### 2.2.3 Face Landmarks

Face landmarks are crucial points on the human face. Combined, they can serve as a two or three dimensional model of the current face and its pose.

Fitting a landmark model in the bounding shape that is returned by the face detector is typically the second step in deeper face analysis.

No fully standardized agreement exists on the specific landmark model design. The existing spectrum ranges from models with only a very few points up to ones with several dozen landmarks. For example, Köstinger et al. [AFLW11] used a model with only 21 landmarks, whereas Le et al. [Helen12] provides 194 of them.

Figure 2.5 shows a model with 68 landmarks. This variant was presented by Gross et al. [Multi-PIE08] and has recently drawn attention, because Sagonas et al. [300W13][300W16] updated several previous databases with annotations according to this model.

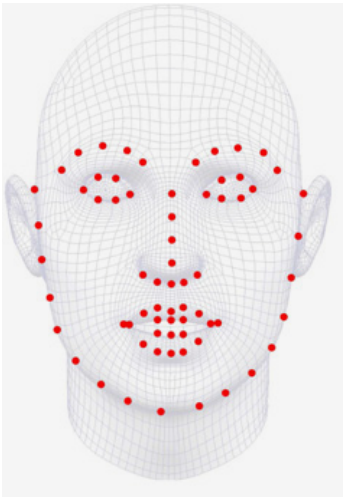

**Figure 2.5:** Face Landmarks

### 2.2.4 Action Units

Ekman and Friesen [FACS78] proposed the first *Facial Action Coding System* (FACS). According to the FACS manual, facial expressions can be deconstructed into more atomic elements called *Action Units* (AU).

Action Units are based on the real actors - the face muscles. FACS describes in detail, which muscles and AU combinations are involved in the different facial expressions.

Figure 2.6 shows *Zygomaticus major* (AU12, Lip Corner Puller) in red. This is the dominant indicator for a happy facial expression. Another typical example would be *Depressor anguli oris* (AU15, Lip Corner Depressor) for sadness.

Some Action Units, such as AU1, can be involved in several different facial expressions and hence must be interpreted in context of others to allow meaningful implications. Furthermore, AU51-56 describe the head pose in yaw, roll and pitch angles. AU45 signals eye blinking.

An excerpt of defined AU with a strong relation to emotions is presented in table 2.1 on page 25.

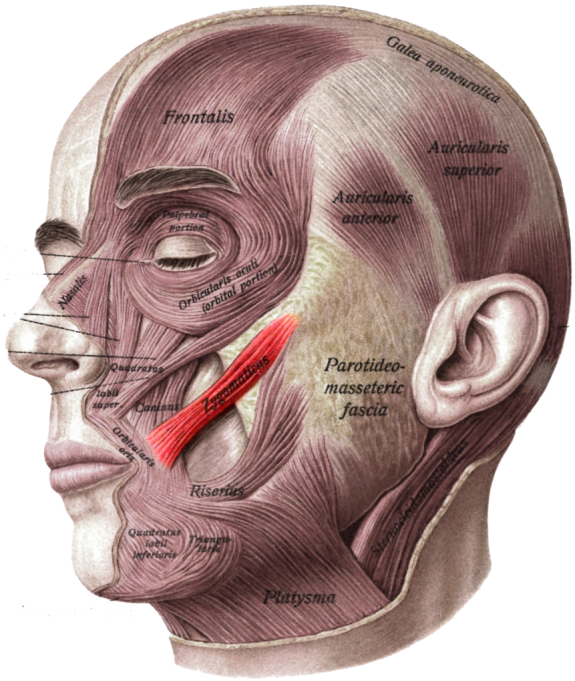


**Figure 2.6:** Face Muscles

FACS and AU became of great interest for computer scientists, because they allow the inference of emotions and serve nicely as advanced input feature space in machine learning classification. The idea is to detect and extract AU values from the face first, instead of applying emotion classification on a rather unsuitable feature space (e.g. full image).

| AU | Name | Happy | Sad | Surprised | Feared | Angry | Disgusted |
|---|---|---|---|---|---|---|---|
| 1 | Inner Brow Raiser | - | X | X | X | - | - |
| 2 | Outer Brow Raiser | - | - | X | X | - | - |
| 4 | Brow Lowerer | - | X | - | X | X | - |
| 5 | Upper Lid Raiser | - | - | X | X | X | - |
| 6 | Cheek Raiser | X | - | - | - | - | - |
| 7 | Lid Tightener | - | - | - | X | X | - |
| 9 | Nose Wrinkler | - | - | - | - | - | X |
| 10 | Upper Lip Raiser | - | - | - | - | - | - |
| 12 | Lip Corner Puller | X | - | - | - | - | - |
| 14 | Dimpler | - | - | - | - | - | - |
| 15 | Lip Corner Depressor | - | X | - | - | - | X |
| 16 | Lower Lip Depressor | - | - | - | - | - | X |
| 17 | Chin Raiser | - | - | - | - | - | - |
| 20 | Lip Stretcher | - | - | - | X | - | - |
| 23 | Lip Tightener | - | - | - | - | X | - |
| 26 | Jaw Drop | - | - | X | X | - | - |

**Table 2.1:** Action Units and Related Emotions

**Figure 2.7:** AU02[1]

**Figure 2.8:** AU04[1]

**Figure 2.9:** AU12[1]

**Figure 2.10:** AU15[1]

**Figure 2.11:** AU23[1]

**Figure 2.12:** AU26[1]

[1]Images were taken from https://www.cs.cmu.edu/~face/facs.htm

### 2.2.5 Face Databases

Selecting the right samples is a critical step for successful machine learning classification. Thus, it's not surprising that a considerable amount of human face databases have been crafted and released to the research community over the last decades.

Early databases, such as *AT&T*, *Yale A* and *JAFFE*, were originally rather byproducts of scientific work that required a set of face images to operate on. These early pioneers contain only a few hundred low-resolution gray scale images taken under laboratory conditions. Figure 2.13 shows subject 5 (happy) from the early *Yale A* database. Later on, more comprehensive databases, like *MMI*, *LFW*, *CK+* or *DISFA*, were explicitly engineered with the intent to provide researchers an advanced base for building and testing their face related classifiers on.

**Figure 2.13:** Yale DB

**Figure 2.14:** Helen DB

Based on their primordial purpose, the properties of these databases can differ significantly. Some of them contain only perfectly aligned faces showing voluntary facial expressions in optimal lighting conditions. Others are often referred to as *in-the-wild* datasets, providing thousands of faces in all kind of natural poses and with possible occlusions of the face. Figure 2.14 shows an example from the *Helen in-the-wild* database. Most of the databases are free for research purposes and can be either directly downloaded or at least after signing and sending an EULA to the authors.

Finally, some databases like *CK+* or *Multi-PIE* were assembled with a high grade of semantical annotations, while others are limited to the plain images only. The possible annotations range from landmark positions to facial expression labels and even action unit intensities.

For an excerpt of existing face databases please refer to Table 2.2 on page 27.

| Database | Year | Images | Subjects | Resolution | Free | In-the-wild | Ref |
|---|---|---|---|---|---|---|---|
| AT&T | 1992 | 400 | 40 | 92x112 | Yes | No | [ATT92] |
| Yale A | 1997 | 166 | 15 | 320x243 | Yes | No | [YaleA97] |
| JAFFE | 1998 | 213 | 10 | 256x256 | Yes | No | [JAFFE98] |
| Yale B+ | 2005 | 16128 | 28 | 640x480 | Yes | No | [YaleB+05] |
| MMI | 2005 | | | | | | [MMI05] |
| BU-3DFE | 2006 | | 100 | 640x480 | Yes | No | [BU-3DFE06] |
| LFW | 2007 | 13233 | 5749 | 250x250 | Yes | Yes | [LFW07] |
| Multi-PIE | 2008 | >750000 | 337 | | No | | [Multi-PIE08] |
| CK+ | 2010 | 593 | 123 | 640x490 | Yes | No | [CK+10] |
| MUG | 2010 | | 86 | 896x896 | Yes | No | [MUG10] |
| AFLW | 2011 | | | | | | [AFLW11] |
| Helen | 2012 | 2330 | | | Yes | Yes | [Helen12] |
| DISFA | 2013 | 4845 | 27 | 1024x768 | Yes | No | [DISFA13] |

**Table 2.2:** Popular Databases with Face Images

## 2.3 Biosensors

### 2.3.1 Electroencephalogram

The *electroencephalogram* (EEG) was invented in 1924 by a German psychiatrist named Hans Berger. This device captures wave activity from several electrodes distributed on the skull. For further details about its invention, please refer to Gloor [EEG69]. An EEG is sometimes also referred to as a *Brain-Computer-Interface* (BCI).

Available EEG models in the market range from rather cheap ones for hobbyists up to expensive ones for medical and research purposes.

The former ones, for instance the *InteraXon Muse*, are priced at several hundred dollars and provide only a few electrodes (points of measurement). The latter ones, for instance the *Mitsar EEG*, can cost up to several thousand dollars and may have over 20 electrodes. Finally, there are some midrange EEG, for instance the *Emotiv EPOC* shown in Figure 2.15, which offers a compromise between price and available features.


**Figure 2.15:** Emotiv EPOC

In the past, EEG devices have been used to successfully extract certain information from brain waves. Hence, besides the raw electrode signals, most EEG also provide some enhanced outputs based on internal algorithms:

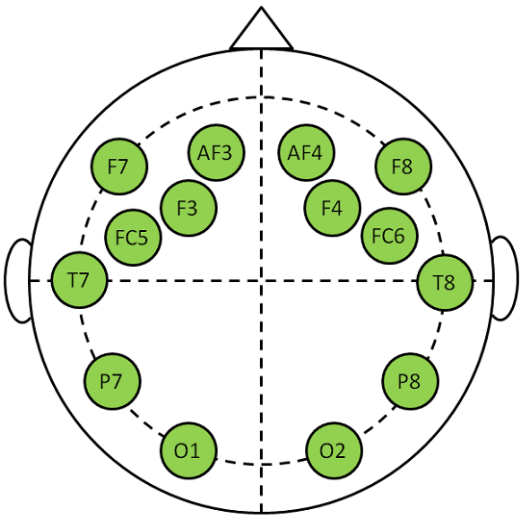


**Figure 2.16:** Emotiv EPOC electrodes

For instance, the *Emotiv EPOC* provides the data below:

- **Motion Detection**: Accelerometer, Gyroscope, Magnetometer
- **Facial Expressions**: Eye-Blink, Smile, Clench, Laugh, Furrow/Raise Brow
- **Moods**: Valence, Engagement, Frustration, Meditation, Excitement, Focus

### 2.3.2 Heart Rate Monitor

*Heart Rate Monitors* (HRM) are devices which recognize contractions of the heart muscle. HRMs typically measure the *R-R interval*, the time difference between consecutive heart beats and forecast the *Beats per Minute* (BPM) accordingly.


**Figure 2.17:** Polar H7 HRM

Note that a BPM of 60 does not implicate a fixed *R-R interval* of 1/60s, instead the actual intervals may vary.

Most of the modern HRM devices, such as the *Polar H7* shown in Figure 2.17, operate wireless using *Bluetooth LE* and are mounted on the chest using an elastic belt. They became rather popular thanks to many available fitness apps that can make use of them.

### 2.3.3 Galvanic Skin Response

*Galvanic Skin Response* (GSR) devices measure the conductance of the skin. An increasing level of sweat on the skin increases the conductance. The two electrodes to measure the conductance are typically placed on the inner side of the fingertips.

**Figure 2.18:** eSense Skin Response

Sweating is controlled rather unconsciously by the nervous system and increased sweating can indicate arousal caused by quite different reasons, such as stress but also sympathy.

GSR devices, such as the *eSense Skin Response* device shown in Figure 2.18, usually operate in the unit *microsiemens*.

## 2.4 Frameworks and Libraries

An excerpt of existing computer vision and machine learning frameworks is presented in this section. The focus is on frameworks which are able to execute face related operations, perform classification or otherwise provide access to biosensor data. Furthermore, some of the presented software solutions are related or used later within Chapters 3 or 4.

### 2.4.1 Computer Vision and Machine Learning

Computer Vision Frameworks aim to provide a rich set of basic tools and reusable functionality to researchers and developers. The libraries presented in this subsection often cover overlapping core functionality and can replace each other partially. The provided functionality usually covers at least basic vector and matrix types with their corresponding operations, as well as common algorithms, such as rescaling, filtering or converting images. Some of the presented candidates extend this base set functionality by providing also edge-, feature-, and object-detection algorithms out of box.

#### 2.4.1.1 OpenCV

The *Open Source Computer Vision Library* (OpenCV) was originally initiated by Intel in 1999 and the latest version 3.1 is currently maintained by a foundation named Itseez [OpenCV15]. The current license is the BSD license. OpenCV is written in C++ and is available for all major native platforms providing a C++ compiler, including mobile devices such as Android.


The library provides all kind of helpful tools. This includes highly dynamic vector and matrix classes with all their implemented operators, as well as typical image conversion functions, such as RGB to gray scale conversion, or mathematical operations (e.g. DFT).

The library was created with the aim of providing not only an open source, but also a highly optimized computer vision framework. It received support for several hardware optimizations over the years, including *CUDA* and *OpenCL*.

OpenCV also provides an implementation of the face detector that was presented by Viola and Jones [ViolaJones01] and is described in 2.2.1.

#### 2.4.1.2 Dlib

Development of *Dlib* started in 2002 and it was finally announced by King [dlib09] as new machine learning toolkit licensed under the Boost license.

Comparable to OpenCV, the library comes with the full set of required base functionality, such as vector, matrix and image classes.

*Dlib* is fully cross-platform compatible and supports all major platforms, such as *Windows*, *Mac* or *Android*.

*Dlib* contains an implementation of the face detector presented by Dalal and Triggs [HOG-SVM05] and mentioned in 2.2.1. Additionally *Dlib* has its own landmark detection implementation and also offers generic *SVM* learning and classification algorithms.

#### 2.4.1.3 Libsvm

The *libsvm* library was presented by Chang and Lin [libsvm11].

It is a state-of-the-art machine learning toolkit that fully focuses on *Support Vector Machines* (see 2.1.2). It supports classification (SVC) and regression (SVR) with all popular kernel types including *linear*, *polynomial*, *radial* and *sigmoid*.

*Libsvm* is fully open source and licensed under a modified BSD license. This makes *libsvm* usable in all kind of cases, including commercial ones.

The library is very easy to integrate thanks to the sources being shipped in a single file. It is well documented, popular and it comes with a full set of command-line tools that allows exploring all features directly.

### 2.4.2 OpenFace

The *OpenFace* library was presented by Baltrušaitis, Robinson, and Morency [OpenFace16]. It is an open source and *state-of-the-art* face analysis framework written in C++11 and is free of charge for research purposes. *OpenFace* contains implementations based on ideas from several earlier publications of the same research group. For further details please also see [CLMZ12][CLNF13][CCNF14].

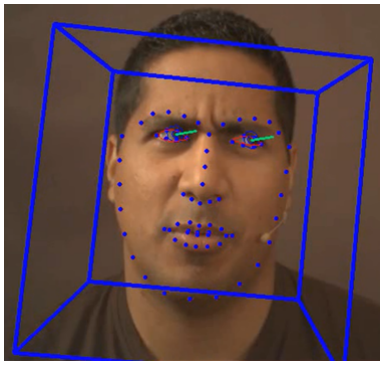

**Figure 2.19:** Visualization of OpenFace Result

*OpenFace* requires the following third party libraries:

- **OpenCV**: Main component for all internal algorithms. Vector, matrix and image processing classes from OpenCV are used. The face detector can be optionally activated instead of the Dlib one.
- **Dlib**: Only for initial face detection and for *HOG* feature extraction and visualization. See 2.4.1.2
- **boost**: Only small parts are required for path related file-system operations.
- **TBB**: Speedup by executing some loops in parallel.

The main features of *OpenFace* are:

- **Face Landmarks and Head Pose Estimation:** *OpenFace* uses a 3D *Point Distribution Model* (PDM) with 51 inner and 17 outer points. Fitting the landmarks into an initial bounding box, which is returned by a common face detector, is done by the *Conditional Local Neural Fields* (CLNF) model, which uses CCNF patch experts to evaluate the likelihood of each landmark. A CLNF model is roughly an improved *Constrained Local Model* (CLM). Using this 3D model, *OpenFace* is able to output a 3D head center position in the camera coordinate system and can also provide the current yaw, pitch and roll angles of the head.
- **Action Unit Extraction:** *OpenFace* provides SVM based classification and regression values for an overall of 17 Action Units. This also includes an AU45 (eye-blink) detector. The AU classifiers are built using a linear kernel SVM and operate on extracted HOG features of the face. The user can select between dynamic and static models for each AU or stay with the defaults.
- **Eye-Gaze Estimation:** *OpenFace* uses additional 3D eye-models. The spheres represented by the model can be used to calculate current eye-gaze estimation vectors. Those are directional vectors which combined with the iris model location define a 3D *line-of-sight* ray. However, the accuracy is limited.

### 2.4.3 Alternative Face Analysis Frameworks

A great number of other commercial and non-commercial frameworks specialized on facial data extraction have been proposed in the last years. In this subsection, a quick overview of some available solutions of different kinds is presented. The sheer mass of possible solutions make it impossible to present them all here.

Existing solutions can be separated into two major types with regard to their basic architecture:

The first major type can be referred to as the *cloud-based* one. All available frameworks require you to upload your images to a proprietary backend operated by the provider. The actual facial data extraction takes place in the cloud. A request with an attached image is answered with a response containing the extracted facial data, such as face locations in the image, landmark positions or facial expressions. *Cloud-based* solutions free the user from worrying about local computing power. However, new problems arise from limited bandwidth, possible quota, interrupted connections or privacy concerns. Especially in real-time tracking scenarios, *cloud-based* solutions can suffer from unacceptable delay or bandwidth consumption.

The second major type operates *fully local*. No information is uploaded and all algorithms run on the local system. However, this can come with non-negligible computing power requirements with regard to mobile or embedded devices. It forces developers to carefully limit the solution to only sparsely selected functionality and to write highly efficient code.

Several different solutions, which were explored as part of the thesis, are described below:

- **clmtrackr**[1]: A JavaScript based facial data extraction framework for browsers available since 2012. Uses the modern *Constrained Local Models* (CLM) approach for landmark fitting. Also provides emotion recognition for *Happy*, *Sad*, *Angry* and *Surprised*. Distributed under the MIT license and completely open source. Based on the work of Saragih, Lucey, and Cohn [clmtrackr11], *clmtrackr* uses several third party libraries, such as the *jsfeat.js* Computer Vision library for JavaScript. The *jsfeat.js* library also provides the cheap *HAAR object detector*, which is a synonym for the work presented by Viola and Jones [ViolaJones01] (see 2.2.1). *clmtrackr* is an excellent choice for web based solutions, however its usefulness for other systems is limited. With regard to embedded devices, such as cars, robots or others, a browser-oriented solution with an unoptimized scripting language seems

[1] https://github.com/auduno/clmtrackr

inappropriate. Even though running in a browser, *clmtrackr* is one of the *fully local* approaches, which are executed by the CPU of the client. The github project has been forked over 700 times so far and several extensions have appeared, which try to add features to the original functionality.

- **Stasm**[1]: A C/C++ implementation based on the well known *Active Shape Models* (ASM) as originally presented by Cootes et al. [ASM95]. ASMs are the oldest kind of the popular landmark model fitting approaches. Several improvements have been made since the original paper, such as the ones included in the latest *Stasm 4.1* by Milborrow and Nicolls [stasm14]. *Stasm* does not come with any further facial data extraction features and is limited to landmark fitting only. Furthermore, *Stasm* internally uses *OpenCV* for image processing and vector math. It's also one of the *fully local* approaches executed by the user's own CPU. It is completely open source and licensed under the BSD license. However, the ASM approach was essentially superseded by *Active Appearance Models* (AAM) and *Constrained Local Models* (CLM) and hence continuing work on *Stasm* might turn into a dead end.
- **CLandmark**[2]: Another C/C++ implementation for facial landmark detection. Based on the work presented by Uřičář et al. [CLandmark15] and the successor of the older and now obsolete *Flandmark* proposed by Uřičář, Franc, and Hlaváč [Flandmark12]. *CLandmark* focuses on fitting a 3D landmark model with the typical 68 points and does not provide any further features. Like many other solutions, *CLandmark* builds upon an existing face detection algorithm. Again it is the [ViolaJones01] implementation in *OpenCV*. *CLandmark* is licensed under GPL 3.0 and hence fully open source.
- **Fraunhofer SHORE**[3]: One of the many available proprietary solutions. The demo version of *SHORE* allows exploring several features, which include detection for faces and face-elements, as well as emotion classification of *Happy*, *Sad*, *Angry* and *Surprised*. Furthermore, *SHORE* attempts to provide age and gender estimation. *SHORE* claims to be fully cross-platform compatible, including mobile device support. Currently, support for the *Google Glass* device is promoted. However, no further platforms except for Windows can be evaluated in the demo and, as usual for proprietary solutions, no detailed information about any internal algorithms or approaches is given. *SHORE* does not seem to fit an actual landmark model, at least none is visualized in the demo or mentioned in the documentation. Hence, it's likely no detailed face landmark positions and face orientation are available.

[1] http://www.milbo.users.sonic.net/stasm/
[2] http://cmp.felk.cvut.cz/~uricamic/clandmark/
[3] https://www.iis.fraunhofer.de/en/ff/bsy/tech/bildanalyse/shore-gesichtsdetektion.html

Like the others so far, *SHORE* is a *fully local* approach executing on the user's own CPU.

- **Sightcorp Crowdsight**[1]: Another popular solution. Supports all major platforms including *Windows*, *Linux*, *Mac*, *iOS* and *Android*. One of the most feature-rich and promising framework out there, yet a fully proprietary and commercial solution. Supports multiple concurrent targets. Includes facial expression detection for *Happy*, *Sad*, *Surprised*, *Angry*, *Disgusted* and *Feared*. Can also extract face position and orientation. Additionally comes with gender estimation and ethnicity detection. On top of the specially tailored applications for the different use cases and platforms, the company also offers a cloud based service API called *F.A.C.E* with comparable functionality. No detailed insight is given on the used algorithms or technical details.
- **Google Vision**[2]: Google has several connections to facial data extraction in their *Google Vision* segment. Every Android phone comes with basic face detection support out of box. Their latest project is the *Cloud Vision API*, which provides face detection, facial landmark extraction, face orientation and mood estimation. In fact the *Cloud Vision API* has a much larger scope than just facial data analysis and can also label many other objects in images. However, the advanced cloud based solutions are fully commercial and proprietary.

[1] http://sightcorp.com/
[2] https://cloud.google.com/vision/

### 2.4.4 Neuromore

*Neuromore*[1] offers software solutions related to biosensors. They aim to support a wide range of different sensors on many platforms. Natively supported are several EEG, HRM and GSR devices. This includes low-priced consumer devices, such as the *Muse EEG*, as well as high-priced professional ones, such as the *EPOC EEG*. Furthermore, the *OSC* network streaming protocol can be used to integrate any external sensor or other input.

*Neuromore Studio* is the main desktop application. It allows to manage *Classifiers*, *State-Machines* and *Experiences*. An *Experience* includes both a *Classifier* and a *State-Machine*. The *Classifier* handles access to sensors and defines the calculations to execute on the received data. The *State-Machine* controls the flow of the running *Experience*. Transitions in the *State-Machine* are triggered either by the user or specific biosensor input. They can cause audiovisual or other kinds of feedback.

Neuromore is powered by a cloud based architecture. All sensor data captured and calculated during a playback of an *Experience* can be marked for upload and hence allow further processing at a later time. This turns the software into a perfect environment for running user studies involving the evaluation of biosensor data.

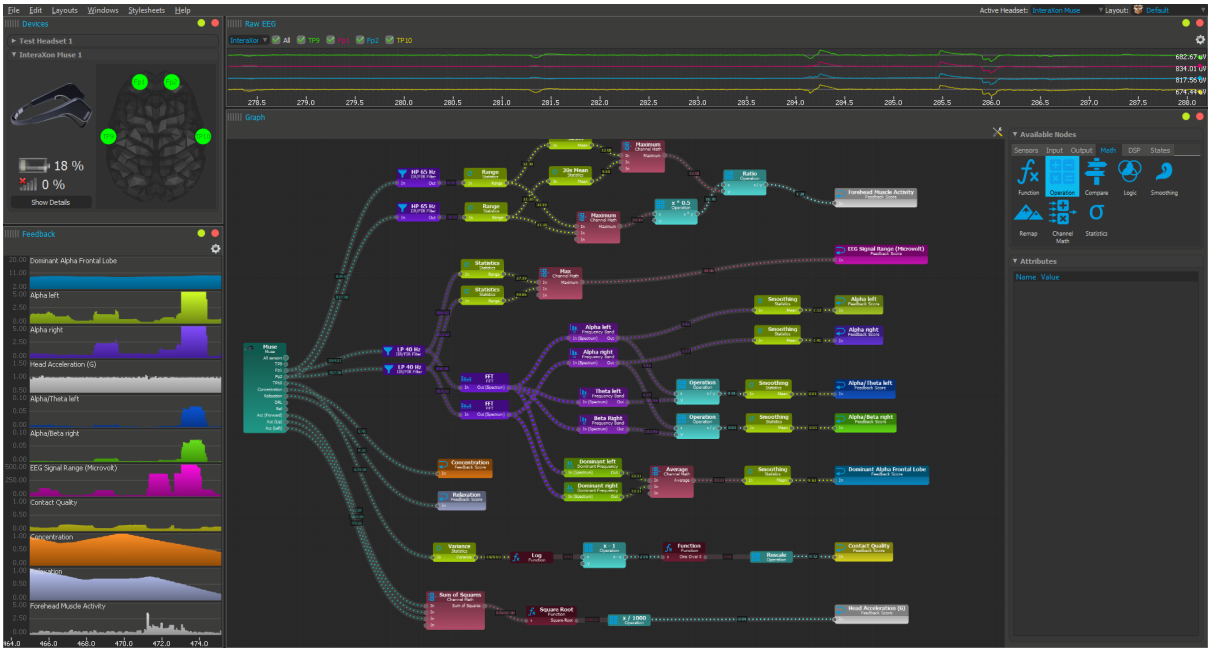

**Figure 2.20:** Classifier in Neuromore Studio

[1] http://neuromore.com

# 3 OpenFace++

In this chapter the library *OpenFace++* and its usage is presented. *OpenFace++* was explicitly created for this thesis and is an essential part of it.

*OpenFace++* is based on the state-of-the-art *OpenFace* library (see 2.4.2), but includes several changes and improvements on top of it. Reinventing the wheel is typically not a desired approach. Speaking in terms of facial data extraction, *OpenFace* has turned out to be a perfect base to improve instead of starting from scratch again. This is mostly due to the facts that *OpenFace* is state-of-the-art, fully open-source, runs locally, is well documented and implemented in fast C++.

The only solutions that seemed to be able to challenge *OpenFace* with regard to quality and functionality are either proprietary, commercial or both (see 2.4.3). Baltrušaitis, Robinson, and Morency [OpenFace16] describe this unsatisfying situation with regard to available open source solutions in their paper as following:

> *Over the past years there has been a huge amount of progress in facial behavior understanding. However, there is still no open source system available to the research community that can do all of the above mentioned tasks (...). There is a big gap between state-of-the-art algorithms and freely available toolkits. This is especially true if real-time performance is wanted - a necessity for interactive systems.*

*OpenFace++* aims to close this gap further by extending the functionality of the original *OpenFace*.

## 3.1 Comparison to OpenFace

*OpenFace++* uses the same internal algorithms for landmark fitting, *Action Unit* (AU) extraction and *Eye-Gaze* estimation as *OpenFace* does. The goal of *OpenFace++* is to add additional features on top of *OpenFace*, to improve its usability, performance and cross-platform capabilities.

The following outlines the benefits of *OpenFace++* over *OpenFace*. For more details please refer to the specific sections of this chapter:

- **Emotion Classification (see 3.5)**: Two different approaches to infer emotions from *Action Units* are implemented in *OpenFace++* . The first one operates using logical expressions on the AU SVM outputs provided by *OpenFace*. The second one is a libsvm powered machine learning approach based on the AU SVR intensity outputs of *OpenFace*.
- **Attention Estimation (see 3.6)**: *OpenFace++* uses the *Eye-Gaze* vectors provided by *OpenFace* to calculate an attention scoring of the subject in front of the camera. Maximum attention is defined as looking straight into the camera. Attention is then scaled by the angle between *Eye-Gaze* and this straight direction to the camera.
- **Threading Architecture (see 3.2)**: Unlike *OpenFace*, *OpenFace++* is fully encapsulated by its own cross-platform compatible C++11 based background worker thread. A typical *Consumer and Producer* threading pattern was applied. This approach frees the user from worries about the calling thread being blocked for too long and potential lags in the user interface or other components, which share a thread with *OpenFace*. It also improves performance slightly, since the internal thread does not have to execute anything else but the internal algorithms.
- **Simplified API (see 3.7)**: *OpenFace++* comes with a drastically simplified *Application Programming Interface* (API) for the user. The original *OpenFace* requires the user to instantiate several different objects. They must be hooked up with each other and they provide lots of unnecessarily public functions irritating the user. In contrast to this, *OpenFace++* requires the user to create only a single, central *FaceSystem* instance with only a handful of intuitively understandable public functions.
- **Sources and Optimized Static Linking (see 3.3)**: The original *OpenFace* comes with precompiled and dynamically linked dependencies for *Microsoft Windows* only. For instance, *OpenCV*, *boost* and *TBB* are all provided this way. These circumstances are bad for cross-platform compatibility and they are not perfect with regard to achieving the maximum performance, which is crucial given the non-negligible computation power requirements. *OpenFace++* fixes these flaws by including the

necessary source-code of all its dependencies, by allowing to easily build them all for different platforms and by linking all the dependencies as static libraries to allow better and adjusted compiler optimization across them. This also ensures that a homogeneous building environment is used for the whole application and all its dependencies.

- **Cross-Platform and Android Support (see 3.4)**: The original *OpenFace* supports only *Microsoft Windows*. However, the presented *OpenFace++* here works on *Microsoft Windows* and *Google Android* out of box. Support for more platforms, such as *Linux* or *Mac*, can be added without much additional work, the bulk of which is setting up the required build chain for compilation. All used dependencies are fully cross-platform compatible.
- **Performance and Memory**: Memory usage from new features is offset by improvements to the original code. For instance, several unused class members have been removed from the original implementation. Additionally, some performance optimizations have been made. But this point is clearly subject to further improvements.

## 3.2 Threading Model

The original *OpenFace* library comes with non-negligible CPU requirements. While already making use of loop parallelization techniques, such as the *parallel_for* statement from Intel's *Threading Building Blocks* (TBB) library, a significant amount of code is still executed from the thread calling the *API* methods and this caller is blocked until facial data extraction has finished for the current frame.

With respect to the notable computational cost and the delay it can cause, this situation is not appropriate. Either each application would have to wrap the *OpenFace* library into a dedicated thread on its own, or the application may run into severe problems with high delays in other tasks which the invoking thread has to perform as well. For instance, this may cause the user interface to lag.

To free users from these worries, the presented *OpenFace++* library comes with its own dedicated thread to perform all work internally. The implementation uses the *::std::thread* class, which is part the official C++11 standard, and hence is fully cross platform compatible.

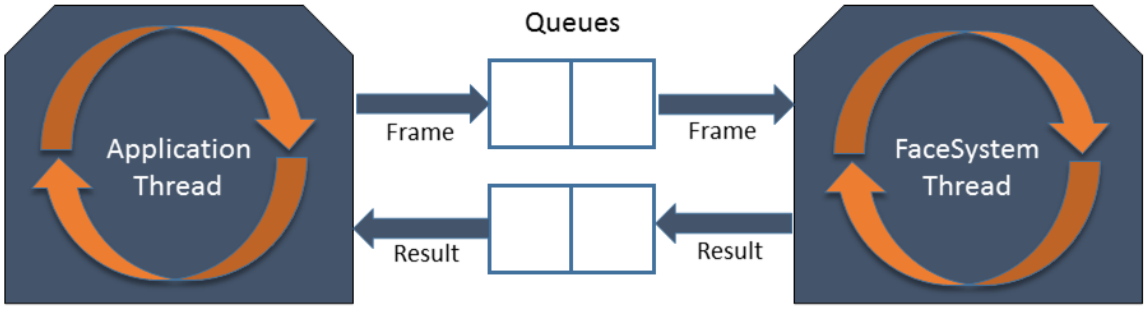


**Figure 3.1:** OpenFace++ Threading Model

A typical *Consumer and Producer* pattern was implemented in *OpenFace++* and is visualized in Figure 3.1. The pattern and figure are described below:

The *Application Thread* is a producer of new *Frames*. A *Frame* is a single image captured from camera or read from disk. The new *Frame* is added to the *Frames-Queue* and later consumed by the *FaceSystem Thread*. Consuming a *Frame* in the *FaceSystem Thread* includes executing facial data extraction and producing a *Result* for this *Frame*. The produced *Result*, which also has the processed *Frame* attached, is then added to the *Results-Queue* and at some point later consumed by the *Application Thread*.

The lockable queues are built upon *::std::queue* and *::std::mutex* classes. Both fully comply with C++11 standards. The queues are used to provide safe inter-thread communication in both directions.

Each of the queues provides two slots. This allows up to a small tolerance on the execution time of the worker when processing a frame. Especially an initial frame can have an increased processing time due to the additional initialization steps, such as face detection. The two slots allow the system to be a bit late on one frame, but to make up for it on the next one and hence not cause any missed frames at all. Adding more slots to the queues increases the maximum possible delay between capturing a frame and receiving its analysis results back and hence is not recommended. For constantly overloaded systems it would even make the situation worse.

## 3.3 Dependencies and Linking

*OpenFace* and *OpenFace++* both come with several third party dependencies, yet there are some significant differences between the two variants. *OpenFace++* builds its required dependencies 100% from compatible sources which are provided together with *OpenFace++*, while *OpenFace* relies on precompiled libraries for *Microsoft Windows* only. Additionally, *OpenFace++* builds all dependencies into static libraries with maximum optimization enabled. Finally, the *boost* dependency has been removed in *OpenFace++*. This decision was made because *boost* is a rather unhandy big framework and only a very few parts of it were actually used.

*OpenFace++* makes use of the following third party libraries:

- **OpenCV**: Like in *OpenFace*, it is the main component for all internal algorithms. Vector, matrix and image processing classes from OpenCV are used. For more details about *OpenCV*, please refer to 2.4.1.1.
- **Dlib**: Only for initial face detection and for *HOG* feature extraction and visualization. More information about *Dlib* can be found in 2.4.1.2.
- **TBB**: Speedup by executing some loops in parallel.
- **libsvm**: A new dependency of *OpenFace++*. This library is a state-of-the-art *Support Vector Machine* (SVM) learning toolkit (see 2.4.1.3). Used for emotion classification.

## 3.4 Cross-Platform and Android Support

Another great improvement of the presented *OpenFace++* library is its implementation with cross-platform compatibility in mind. It is strictly built using C++11 only and hence is free of any platform specific API calls. These characteristics also support and encourage porting *OpenFace++* to more platforms.

The original *OpenFace* could only be used on *Microsoft Windows*. In contrast to this, the presented *OpenFace++* additionally comes with full Android support out of box. The library and all of its dependencies can also be built using the official *Android NDK 11c* or later.

All Android CPU architectures except for the now obsolete ARM variants before v7a are supported. The minimum supported Android version is *4.4.2* (API-19) for ARM and X86. For ARM64 and X64, at least Android *5.0.1* (API-21) is required.

All necessary tools are part of the thesis attachments. A *Visual Studio 2015* solution (*Engine-Android.sln*) is included there to allow one-click building *OpenFace++* and all of its dependencies for Android.

Please note that any native library must be wrapped by an Android specific *Java Native Interface* (JNI) wrapper to become available in the managed Java world of Android. This JNI wrapper builds *OpenFace++* into a compatible *Shared Object* (SO) library which includes the Java-callable methods and which can then be loaded and used at runtime in the scope of any arbitrary Android App. This wrapper is also part of the thesis attachments and ready to use.

All variants of *OpenFace++*, regardless of the specific platform, provide equal capabilities and hence come with roughly the same hardware requirements:

On a Samsung Galaxy S6 reference phone with eight CPU cores, *OpenFace++* runs at round about 10 FPS while in tracking mode, but can cause small lags when the face is lost and more expensive reinitialization has to be performed. Hardware acceleration through *OpenCL* has not been enabled yet, but might yield further improvements.

It also seems reasonable to switch to a faster but less accurate face-detection algorithm on mobile platforms. For instance, the approach presented by Viola and Jones [Viola-Jones01] from *OpenCV* could be used instead of [HOG-SVM05] from *dlib*. Please refer to 2.2.1 for more details. This is only one possible optimization amongst many to improve support on mobile devices.

The thesis attachments also include an exemplary App project named *OpenFaceAndroid* for the official *Android Studio*. This example demonstrates how to correctly use the built and JNI wrapped *OpenFace++* library with the internal camera of a smartphone.

Due to backward compatibility with Android 4.4.2, the *OpenFaceAndroid* example currently uses the deprecated *android.hardware.Camera* interface and its *Camera.PreviewCallback* to forward the raw pixel data of a camera captured frame to *OpenFace++*.

Figure 3.2 below shows the mentioned *OpenFaceAndroid* App running on a Samsung S6 Galaxy phone which is pointed on a display with a face image opened.

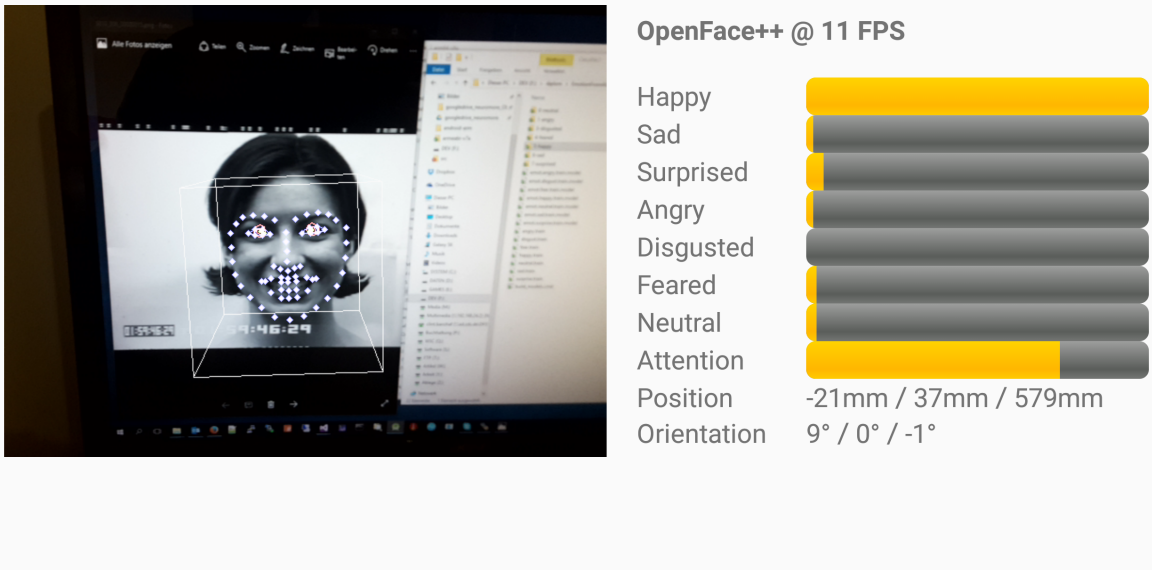


**Figure 3.2:** OpenFace++ on Android Phone

## 3.5 Emotion Classification

Being able to classify human emotions based on extracted *Action Units* (AU) is a significant new feature of the presented *OpenFace++* library. The original OpenFace library (see 2.4.2) is able to extract AU nicely, but it does not provide any kind of inferred emotions. *OpenFace++* includes two emotion classifiers described below.

### 3.5.1 Classifier 1: Logical Expressions

The first classifier examines the boolean AU values. True means the corresponding AU is activated right now. False means the AU is deactivated. Facial expressions are typically a mix of several concurrently activated AU (see 2.2.4). The evaluated expressions are shown in table 3.1. Furthermore, a simple filter is used: An expression must evaluate to true for at least two adjacent frames.

The limitation with this approach is clearly inaccuracy. It results from the rather coarse true or false classification of each AU. This does not cover the grade (e.g. max or low) of the activation.

| **Emotion** | **Expression** |
|---|---|
| Happy | AU06 $\wedge$ AU12 |
| Sad | AU15 $\wedge$ $\neg$AU06 $\wedge$ $\neg$AU12 $\wedge$ $\neg$(AU25 $\wedge$ AU26) |
| Surprised | AU01 $\wedge$ AU02 $\wedge$ AU05 $\wedge$ AU25 |
| Disgusted | AU09 $\wedge$ AU10 $\wedge$ $\neg$(AU06 $\wedge$ AU12) |
| Angry | AU04 $\wedge$ AU07 $\wedge$ AU23 $\wedge\neg$(AU06 $\wedge$ AU12) |
| Feared | AU01 $\wedge$ AU02 $\wedge$ AU05 $\wedge$ (AU07 $\vee$ AU20) $\wedge$ AU25 $\wedge$ $\neg$(AU06 $\wedge$ AU12) |
| Neutral | $\neg$(HAPPY $\vee$ SAD $\vee$ SURPRISED $\vee$ DISGUSTED $\vee$ ANGRY $\vee$ FEARED) |

**Table 3.1:** Logical Expressions for Emotion Classification

### 3.5.2 Classifier 2: Support Vector Regression

The second classifier examines the floating point AU values. A high value means a high grade of activation. The classifier contains seven more internal regression classifiers. These were built using *libsvm* (see 2.4.1.3) and trained on samples from the *CK+* image database (see 2.2.5).

The positive samples have been assigned an output value of 1.0 and the negative ones a value of 0.0. This choice was made to best match the desired output result. The SVM type is *epsilon-SVR* with a *radial kernel*.

For each classifier, one of the seven classes serves as positive sample, while all other serve as negatives. This approach is often referred to as *One-vs.-Rest* or *One-vs.-All*. The advantage of this approach is the increased likelihood that a maximum in one classifier causes minimums in all others. Hence, the likelihood for ambiguous results among the seven classifiers are generally reduced as well.

| AU | Happy | Sad | Surprised | Disgusted | Angry | Feared | Neutral |
|---|---|---|---|---|---|---|---|
| AU01 | -2.067 | +1.073 | **+4.021** | -2.115 | -1.197 | +1.962 | -0.388 |
| AU02 | -0.591 | +0.396 | **+2.183** | -0.971 | -1.471 | +0.946 | -0.181 |
| AU04 | -2.406 | -0.444 | +1.338 | +1.965 | **+1.941** | -0.381 | -1.087 |
| AU05 | -0.954 | -0.067 | **+2.866** | -2.493 | -0.129 | **+2.606** | -1.058 |
| AU06 | **+2.217** | -1.041 | -1.328 | +1.888 | -0.848 | -0.304 | -0.990 |
| AU07 | +2.326 | +0.152 | -0.106 | +4.873 | +0.317 | +0.632 | +0.093 |
| AU09 | +0.243 | -1.111 | -0.300 | **+4.282** | +0.298 | -0.303 | -0.658 |
| AU10 | +1.558 | -0.855 | -0.536 | +0.907 | -1.899 | -0.488 | -1.706 |
| AU12 | **+2.500** | -0.469 | -0.132 | -0.438 | -1.277 | +0.141 | -0.247 |
| AU14 | +1.664 | -1.557 | -1.393 | -0.570 | -0.859 | -0.606 | -1.413 |
| AU15 | -0.598 | **+2.185** | -0.374 | -0.044 | +0.415 | +1.178 | -0.080 |
| AU17 | -0.120 | +1.541 | +0.937 | +1.734 | +1.465 | +0.310 | -0.158 |
| AU20 | -0.329 | +0.410 | -0.953 | -1.011 | **+1.987** | **+2.988** | -0.588 |
| AU23 | +0.313 | -0.128 | -1.162 | +0.507 | +0.944 | -0.556 | +0.106 |
| AU25 | -0.139 | -1.098 | **+2.847** | -0.376 | -1.240 | +1.358 | -0.789 |
| AU26 | +0.557 | +0.071 | +1.640 | +0.980 | -0.136 | +0.574 | -0.133 |
| AU45 | -0.436 | +0.201 | +0.478 | +0.641 | +0.864 | -0.438 | -0.439 |
| **Fig.** | **3.3** | **3.4** | **3.5** | **3.6** | **3.7** | **3.8** | **-** |
| **Samples** | **67** | **16** | **20** | **37** | **21** | **7** | **25** |

**Table 3.2:** Examples of AU SVR Values of Emotions

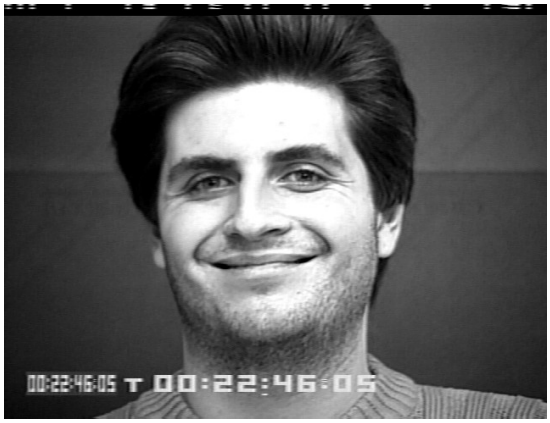

**Figure 3.3:** Happy[1]

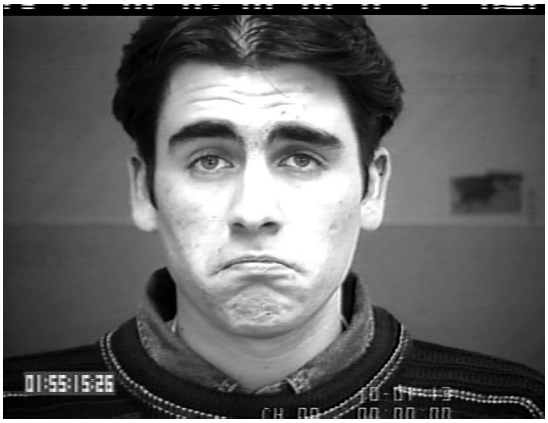

**Figure 3.4:** Sad[1]

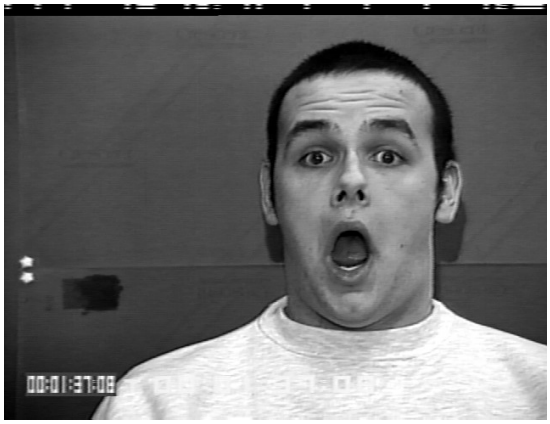

**Figure 3.5:** Surprised[1]

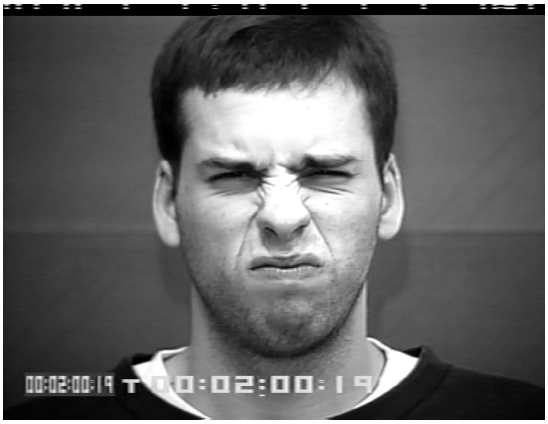

**Figure 3.6:** Disgusted[1]

**Figure 3.7:** Angry[1]

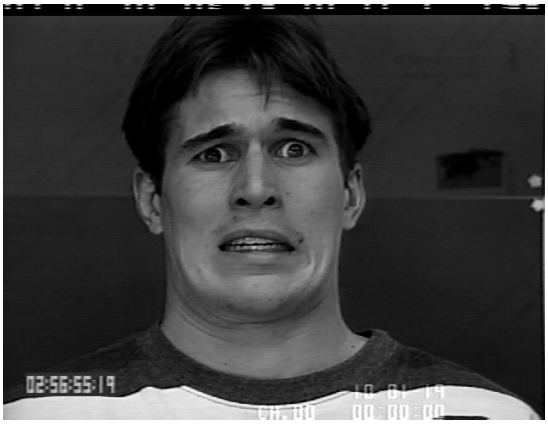

**Figure 3.8:** Feared[1]

[1]Images are part of the CK+ database

### 3.5.3 Limitations

Several limitations exist in the proposed emotion classifications. Below the most important ones are described.

First, as with basically any other classification approach, there is a strong dependency on the reliability of processed input data. Incorrectly extracted *Action Units* can easily cause both classifications to fail. For instance, a beard drastically reduces the quality of extracted AU in faces and hence the emotion classification as well. In this case, it is due to the skin being partially covered and therefore causing fuzzy HOG features in the AU extraction.

Second, *OpenFace* does not provide all *Action Units*. It is missing AU11, AU13, AU16, AU18, AU22, AU24, AU27 and AU28. AU16 in particular is interesting for the classification of disgust, and the others would be helpful in training the SVR classifier. Ideally, this could increase the separation of yet ambiguous responses further.

Third, some facial expressions are quite clear, but sometimes a single frame of a facial expression can be ambiguous even for human emotion recognition. Especially the classes angry and disgust are often difficult to separate. Does figure 3.9 show an angry or disgusted face? The correct answer would probably be both. But using the AU set of such a sample for either angry or disgust in the classifier-training will cause trouble with the mentioned *One-vs.-Rest* approach.

**Figure 3.9:** Ambiguous Facial Expression

Fourth, the current classifications operate on a single set of AU, which are extracted from a single image. Almost no temporal relation is examined or exploited in the classifiers to improve performance in any way, excepting the small filter in the logical classifier. However, facial expressions typically last for several frames and likely also contain phases of increasing and decreasing intensity. Fortunately, this feature can be easily and more dynamically built on top of the current implementation.

Fifth, human emotion recognition can involve more senses than seeing. For instance, one can easily image some typical sounds for happiness (laughing), sadness (crying), anger (growling) or fear (screaming). Especially when it comes to ambiguous facial expressions, these additional hints serve well in distinguishing different emotions, but are absent in the proposed classifiers.

## 3.6 Attention Scoring

*OpenFace++* includes an implementation for *Attention Scoring*. The attention of the subject in front of the camera is inferred from the *Eye-Gaze* vectors.

The origin of the coordinate system is located at the *Webcam*. The *Pupil Position* (P) can be retrieved from the fitted landmark model. The *Point of Interest* (Q), on which the subject is currently focusing, may vary. The vector $\vec{v}$ is defined as the straight ray from P into the camera and the vector $\vec{u}$ is a scaled variant of the *Eye-Gaze* vector extracted by *OpenFace* and pointing from P towards Q. The *Attention Score* is then defined linearly scaled based on the size of the delta angle $\varphi$ between vectors $\vec{u}$ and $\vec{v}$. Maximum attention (1.0) is achieved by looking straight into the camera ($\varphi = 0$). Minimum attention (0.0) is defined by some threshold angle ($\theta$). By default, *OpenFace++* maps $\varphi = \theta = \frac{1}{10}\pi$ (18°) and beyond to zero attention. The math is described below, for a visualization please see Figure 3.10.

$$\varphi = \arccos \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} \tag{3.1}$$

$$attention(\varphi) = \max\left(0.0, \min\left(1.0, 1.0 - \left(\varphi \cdot \frac{1.0}{\theta}\right)\right)\right) \tag{3.2}$$

Note that this approach is obviously sensitive to the distance between pupil and camera. Given a point Q and a camera position, the angle $\varphi$ (and hence also the attention) changes with this distance. The current implementation is adjusted to work best in typical real-world scenarios with distances between 30cm and 70cm. Below, it's likely only parts of the face are captured and above, the extracted *Eye-Gaze* vectors start to become inaccurate.

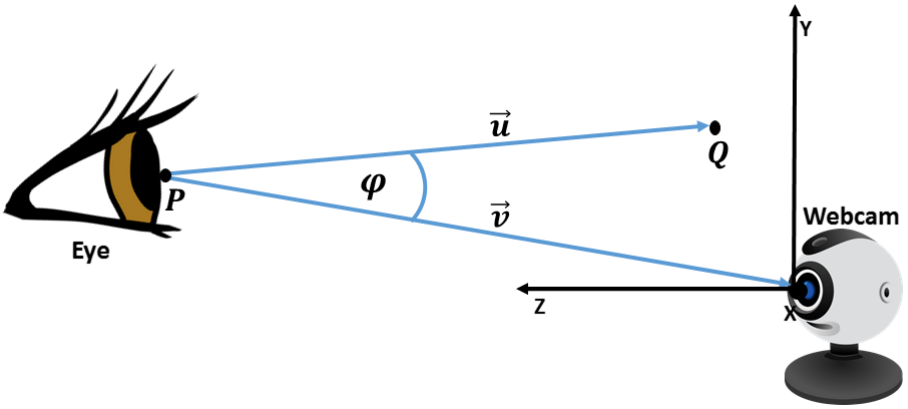


**Figure 3.10:** Attention Scoring based on delta angle.

## 3.7 Using the library

In this section, the API and possible interactions between an application and the *OpenFace++* library are described. The API is kept very simple and can be learned fast, one more improvement of the presented *OpenFace++*.

### 3.7.1 Class: FaceSystem

The *FaceSystem* class is the main component of the *OpenFace++* library. It must be instantiated by providing file locations of necessary data files in the constructor. An instance of *FaceSystem* is all that is necessary to operate the library. For further details please also refer to the example in 3.8.

Once the *FaceSystem* instance has been created, its internal worker thread must be started by invoking the following method on the *FaceSystem* instance with a *true* parameter:

- void setIsRunning(bool isRunning)

Afterwards, the *FaceSystem* instance provides the two operations below to enqueue frames for analysis and to retrieve back available results:

- bool enqueueWork(cv::Mat* item)
- Result* dequeueResult()

Both methods should be called regularly from your application thread. A *false* value returned from *enqueueWork()* means that both queue slots were already filled. In this case you might be constantly providing input frames faster than *FaceSystem* can process them on your system or you have not started the *FaceSystem* yet. In case *dequeueResult()* returns a null pointer, no available result was found in the queue.

Additionally the *FaceSystem* can be fully reset at any time. For instance it should be reset between analyzing completely different face images. The reset can be done by calling below method on the *FaceSystem* instance:

- void reset()

Please be aware that all presented operations are executed fully asynchronously. All of these methods return almost instantly and none will block the invoking thread. Accordingly, the internal worker thread will always finish its current cycle first, before processing a reset or other request.

### 3.7.2 Class: Result

The *Result* class bundles all available analysis results into a single data instance. Such an instance is then returned from the asynchronous worker-thread for each processed frame and can be used in the application to take further actions.

An overview of all available data fields on the *Result* class is shown in Table 3.3 below.

| Type | Name | Description |
|---|---|---|
| bool | `faceDetected` | `True if face was found in frame` |
| double | `certainty` | `Certainty of the detection` |
| double | `modelLikelihood` | `Likelihood of the CLNF model` |
| cv::Vec3d | `position` | `3D position of the face` |
| cv::Vec3d | `orientation` | `3D orientation of the face` |
| Eye | `eyeLeft` | `Info about the left eye` |
| Eye | `eyeRight` | `Info about the right eye` |
| ActionUnitValues | `auSVR` | `Action Units regression values` |
| ActionUnitValues | `auSVM` | `Action Units classifier values` |
| cv::Mat | `image` | `Processed and annotated frame` |
| EmotionBools | `emotions` | `Facial expressions based on SVM` |
| EmotionProbability | `emotionProbability` | `Facial expressions based on SVR` |
| double | `getAverageAttention()` | `Avg. attention of both eyes` |

**Table 3.3:** Available Data on Result

The first thing to check on a received *Result* instance is typically the *faceDetected* property. If this turns out to be *false*, then no further values need to be examined, because the analyzed frame did not contain a face at all. However, if the *faceDetected* flag is *true*, then *certainty* and *modelLikelihood* are measurements for the quality and reliability of the extracted facial data and all other values are populated.

The fields *position* and *orientation* describe the face position and orientation in 3D space. The camera serves as point of origin (0/0/0). The unit of the three *position* components is *millimeter*. The unit of the three Euler angles in *orientation* is *radian*. The X component of *orientation* describes the rotation around the X-axis (Pitch). Others are accordingly for rotation around Y-axis (Yaw) and Z-axis (Roll).

The fields *eyeLeft* and *eyeRight* provide further details about the eyes. This includes the 3D position of the eye, as well as the extracted *Eye-Gaze* vector and an estimated *Attention Score* (see 3.6).

The fields *auSVR* and *auSVM* contain the 17 extracted and raw *Action Unit* values as already included in the original *OpenFace*. The floating point values in *auSVM* can only have the values *0.0* or *1.0*, according to the binary classification they belong to. The values in *auSVR* can have any arbitrary regression value representing the current intensify of the corresponding *Action Unit*.

The field *emotions* provides seven boolean values indicating the state of seven facial expression: *Happy*, *Sad*, *Surprised*, *Disgusted*, *Angry*, *Feared* and *Neutral*. These values represent the output of the *Logical Expression based Classifier* (see 3.5.1). These are non-exclusive and it is possible for multiple facial expressions to be flagged as true.

The field *emotionProbability* contains seven floating point regression values indicating the current probability of the seven mentioned possible facial expressions. The values here range from 0.0 to 1.0 and come from a *libsvm* powered *Machine Learning Regression* (see 3.5.2).

## 3.8 Example

The C++11 code on the next two pages provides a complete minimal example on how to use the *OpenFace++* library. The code will initialize a *FaceSystem* instance using the video stream of a local webcam. The analyzed frames are presented in a common window. All parts belong to a single *main.cpp* file, which can be found along with its required dependencies in the attachments of the thesis.

Only the *FaceSystem.h* header file must be included in a project using the *OpenFace++* library. The corresponding classes can be made visible by using the *::OpenFace* namespace. The library requires several data files to be available, it is recommended to place these into a dedicated folder and define it once.

The main function initializes the webcam stream using OpenCV. It constructs the *FaceSystem* and then enters the main loop. This loop includes forwarding captured frames to the asynchronously working *FaceSystem* and handling available results.

```
/*****************************************************************************/
/************************** OpenFace++ Example ***************************/
/*****************************************************************************/

// includes
#include <FaceSystem.h>

// used namespaces
using namespace ::std;
using namespace ::OpenFace;

#define APPNAME "OpenFace++ Example" // window title etc.
#define RES "../../../../Engine/res" // basefolder for resources

//////////////////////////////////////////////////////////////////////////////
// Main entry point
//////////////////////////////////////////////////////////////////////////////
int main(int argc, char** argv)
{
  // create window
  ::cv::namedWindow(APPNAME, 1);
  ::cv::setWindowTitle(APPNAME, APPNAME);

  // open webcam
  ::cv::VideoCapture cap(CV_CAP_ANY);
  cap.set(CV_CAP_PROP_FRAME_WIDTH, 640);
  cap.set(CV_CAP_PROP_FRAME_HEIGHT, 480);

  // failed to open webcam
  if (!cap.isOpened())
    return -1;

  // create openface++ system
  FaceSystem mSystem(RES);

  // start openface++ system
  mSystem.setIsRunning(true);
```

```
  // enter thread loop
  for (;;)
  {
    // allocate a new mat to enqueue later
    ::cv::Mat* frame = new ::cv::Mat();

    // capture a new frame
    // warning: this blocks until a new frame is available
    // typically limits loops to webcam fps @ ~15-60 fps.
    cap >> *frame;

    // try to forward the new frame to face system
    // if queue is full then analyser is busy, so skip frame
    if (!mSystem.enqueueWork(frame))
      delete frame;

    // try to get last processed frame from face system
    // if available, show and delete it
    if (Result* result = mSystem.dequeueResult())
    {
      ::cv::imshow(APPNAME, *result->image);
      delete result->image;
      delete result;
    }

    // use opencv to process windows message queue
    int key = ::cv::waitKey(1) & 0xFF;
  }

  // normal exit
  return 0;
}
```

**Listing 3.1:** OpenFace++ Example

## 3.9 Face Sensor Application

The *Face Sensor Application* is a small piece of software which was created for the thesis and which uses the presented *OpenFace++* library. It displays the plain webcam stream next to the analyzed frames that include the landmark annotations of *OpenFace*. It also shows the output of both new facial expression detectors, the new attention score estimation as well as a lot of the info which the unmodified *OpenFace* also includes. For instance, the face position, orientation and extracted AU.

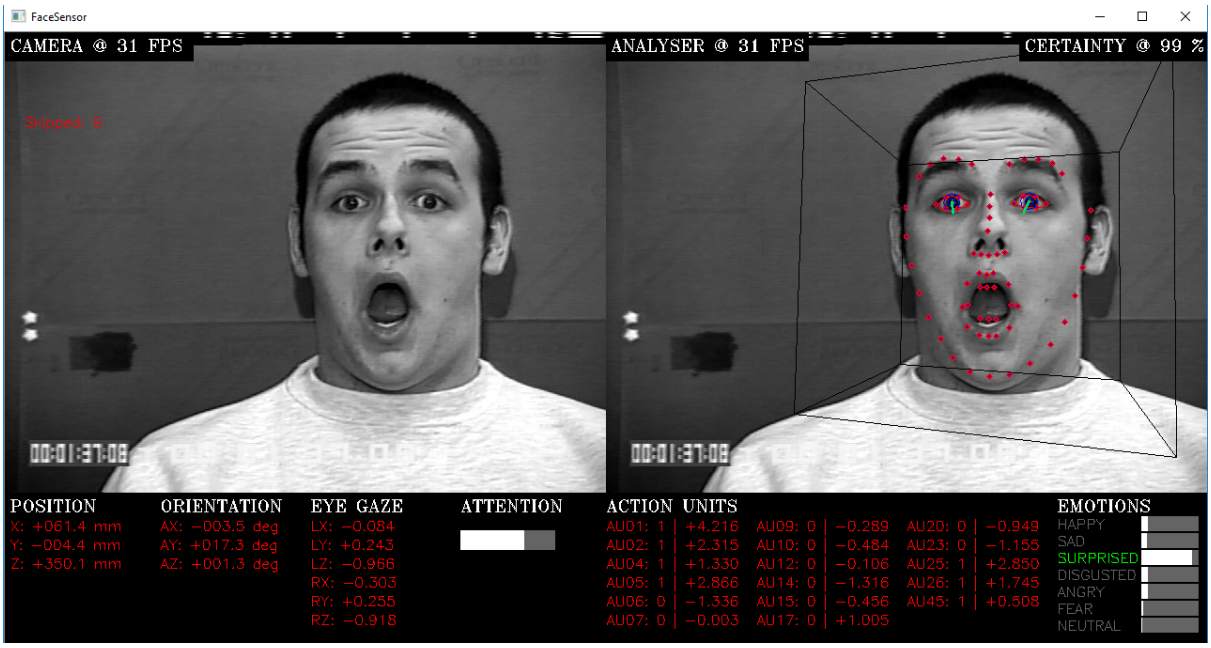


**Figure 3.11:** FaceSensor Application

By default, the *Face Sensor Application* operates on a local webcam stream. However, it is also capable of analyzing PNG image files. To do so, the image must be provided as first argument when starting the executable or simply dragged and dropped on the executable. This feature was also used to capture AU training data sets from face images showing the different facial expressions.

Another feature of the *Face Sensor Application* is the included *OpenSound Control* (OSC) network streaming support. OSC is a lightweight, UDP based data streaming protocol that can be used to forward all received data of *OpenFace++* for more complex analysis on another system.

# 4 User Study

The goal of the study was to investigate the reliability of the extracted facial data as it is provided by the proposed *OpenFace++* and to evaluate the possibility of replacing EEG based detectors for eye-blinking and head motions with webcam based image processing systems. The user study included three experiments which are described below together with the apparatus, details about the participants and the results.

## 4.1 Participants

An overall of eight subjects aged between 26 and 63 took part in the user study. Of these, six were male and two were female. All participants received candies and 5 Euro as compensation. They were made aware of the process and their right to cancel the experiment at any time. All participants signed a consent form agreeing to the terms of the user study.

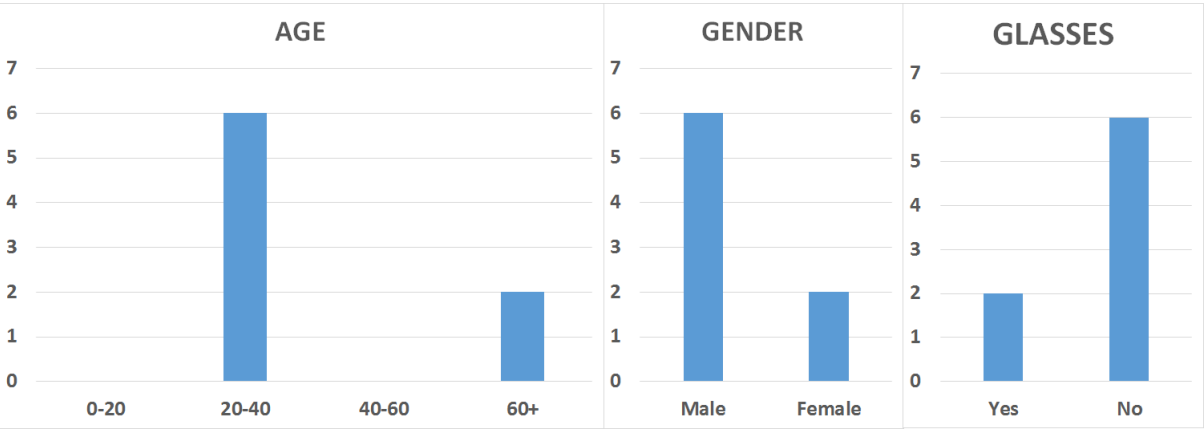


**Figure 4.1:** Statistics of Participants

## 4.2 Apparatus

### 4.2.1 Hardware

First, there are three bio-sensors equipped by the subject. This includes the *Emotiv EPOC EEG* (see 2.3.1), the *Polar H7 HRM* (see 2.3.2) and the *eSense GSR* (see 2.3.3). The *Emotiv EPOC* is used to capture brainwave activity and to evaluate its blink detector and gyroscope in comparison to the performance of *OpenFace++*. The HRM and GSR devices are only used in Experiment 3 to discover possible correlations.

Second, there is the hardware visible to the subject. This includes a flat screen display and an ordinary webcam mounted on top of it.

Third, there is the hardware in the background. This includes a fast Intel i7 notebook, a Samsung Galaxy S6 smartphone and an ordinary access-point providing wireless Internet access to the devices. The smartphone is required to retrieve *HRM* and *GSR* data.

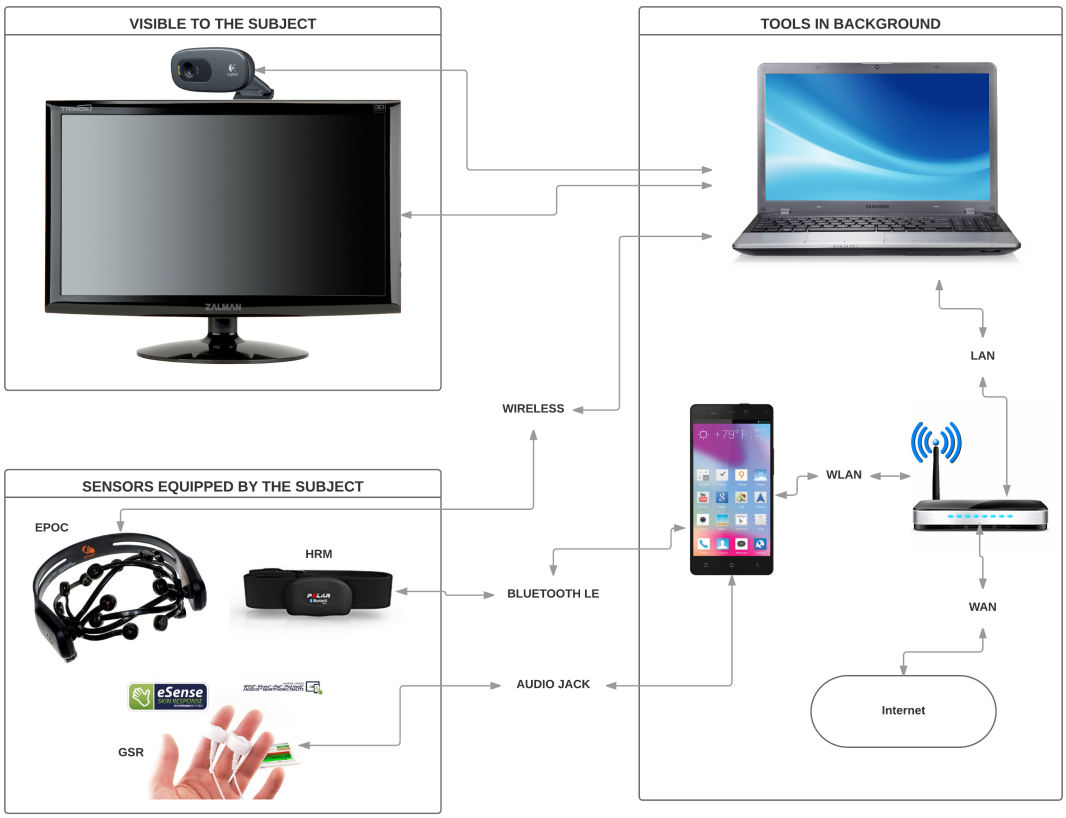


**Figure 4.2:** Schematic Apparatus Hardware

### 4.2.2 Software

Windows 7 64-Bit is installed on the Notebook. The *FaceSensor Application* (see 3.9) is running and processing the frames captured by a webcam. Results of the facial data extraction are streamed to *Neuromore Studio* (see 2.4.4) using the OSC network protocol.

*Neuromore Studio* receives the *Emotiv EPOC* data directly from the device and handles all further playback related parts of the experiments. It triggers audio and visual stimuli as defined in the experiments.

Furthermore, *Neuromore Studio* writes all captured data to *comma-separated values* (CSV) files that can be imported to Excel for statistics later.

### 4.2.3 Physical Installation

The subjects sit centered in front of the flat-screen display and its mounted webcam. The distance between webcam and the face of the participants was measured to be approximately 50cm, but was not perfectly calibrated or unified for each subject or run. Portable, white dividing walls were placed right behind the subject and the display.

**Figure 4.3:** Photo of the Physical Installation with Subject

## 4.3 Experiment 1

The first experiment evaluates the eye-blink detection capabilities of *OpenFace++*. *OpenFace++* and the original *OpenFace* include a binary classification as well as a regression value for the current closeness state of the eyes. Both are fully extracted from captured frames. Additionally, the measured reliability is compared against the performance of alternative approaches involving much more expensive BCI hardware, which typically also provides eye-blink detection.

### 4.3.1 Hypothesis

Image processing based eye-blink detection performs reliably and equally or even better than a comparable, but much more expensive BCI driven approach.

### 4.3.2 Task

The subject is requested to close and open the eyes on command. The stimuli to close or respectively open the eyes occur alternating in fixed intervals of 3 seconds with 5 iterations. The stimuli are audio-visual. The text *Open* or *Close* is shown on a display and an according voice command sounds from the speakers. To keep the results free from any not commanded eye-activity, the subject is asked to ideally suppress any other reflexes to eye-blink. One trial run is granted to get familiar with the task.

### 4.3.3 Independent Variable

The first controlled variable in the experiment is the detection system, namely the *OpenFace++* library or the *Emotiv EPOC* BCI. The second controlled variable is the closeness state of the eyes which is switched by the according stimuli.

### 4.3.4 Dependent Variable

The measured variable is the detection performance of the different approaches.

### 4.3.5 Neuromore Classifier

The classifier of Experiment 1 uses the *Emotiv EPOC* device, two parameters and four extracted facial data values from *OpenFace++*. The first parameter is the reference value *EYES-CLOSED*. The second parameter is *LOOPCOUNT*. The extracted facial data values are *DETECTED*, *CERTAINTY*, *CLOSED BOOL*, and *CLOSED FLOAT*. The raw EEG data is captured together with eye-related classifications, such as *Blink*. All data is written to CSV. Operates at 32 Hz.

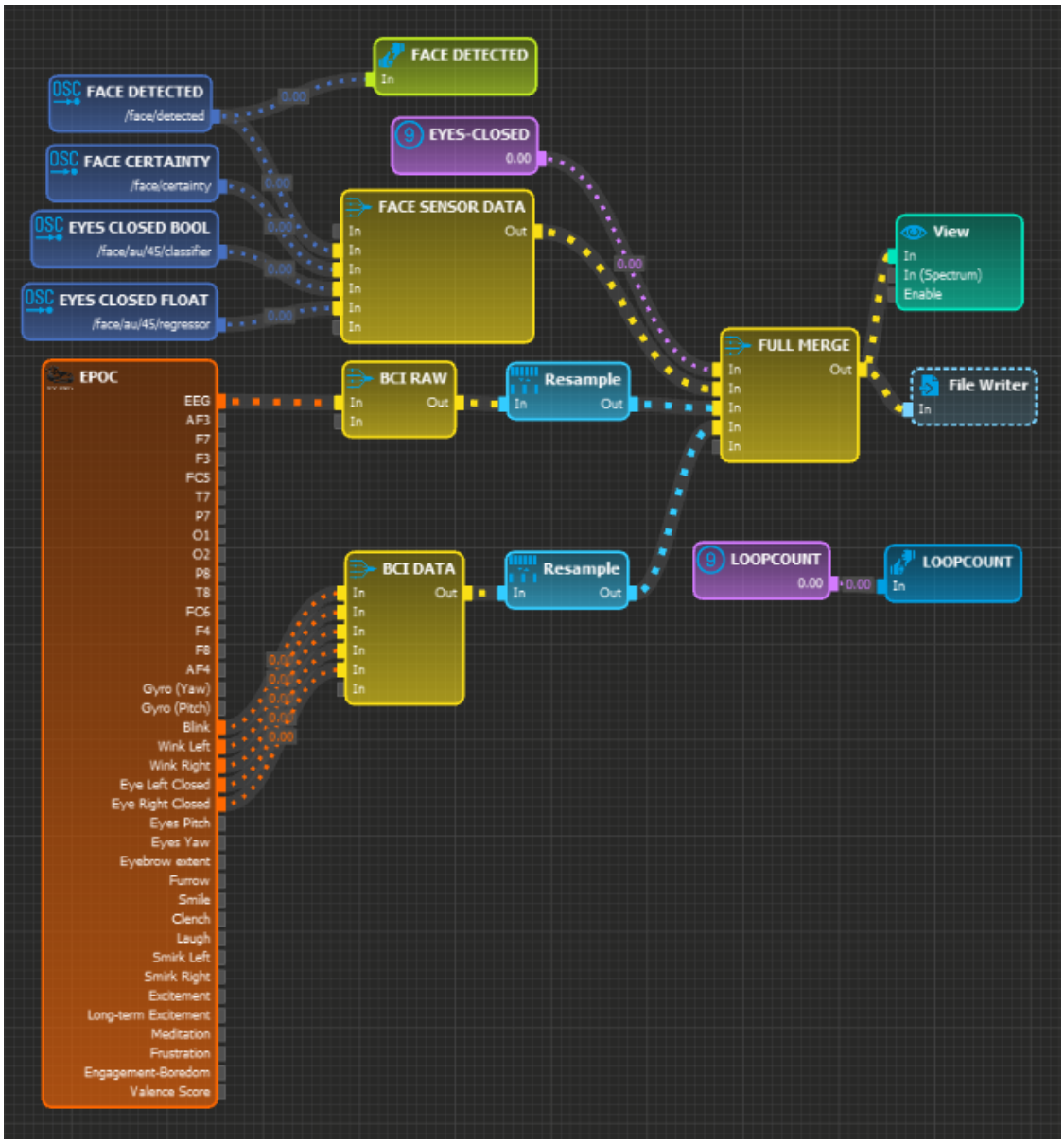


**Figure 4.4:** Classifier of Experiment 1

### 4.3.6 Neuromore State-Machine

The state-machine of Experiment 1 includes four internal states. Entering the *CLOSE* and *OPEN* states triggers the according audio and visual stimuli and flips the *EYES-CLOSED* parameter in the classifier. The *LOOPCOUNT* parameter is evaluated in the *OPEN* state to determine when the experiment has reached its required iterations and the path towards the *END* state should be chosen.

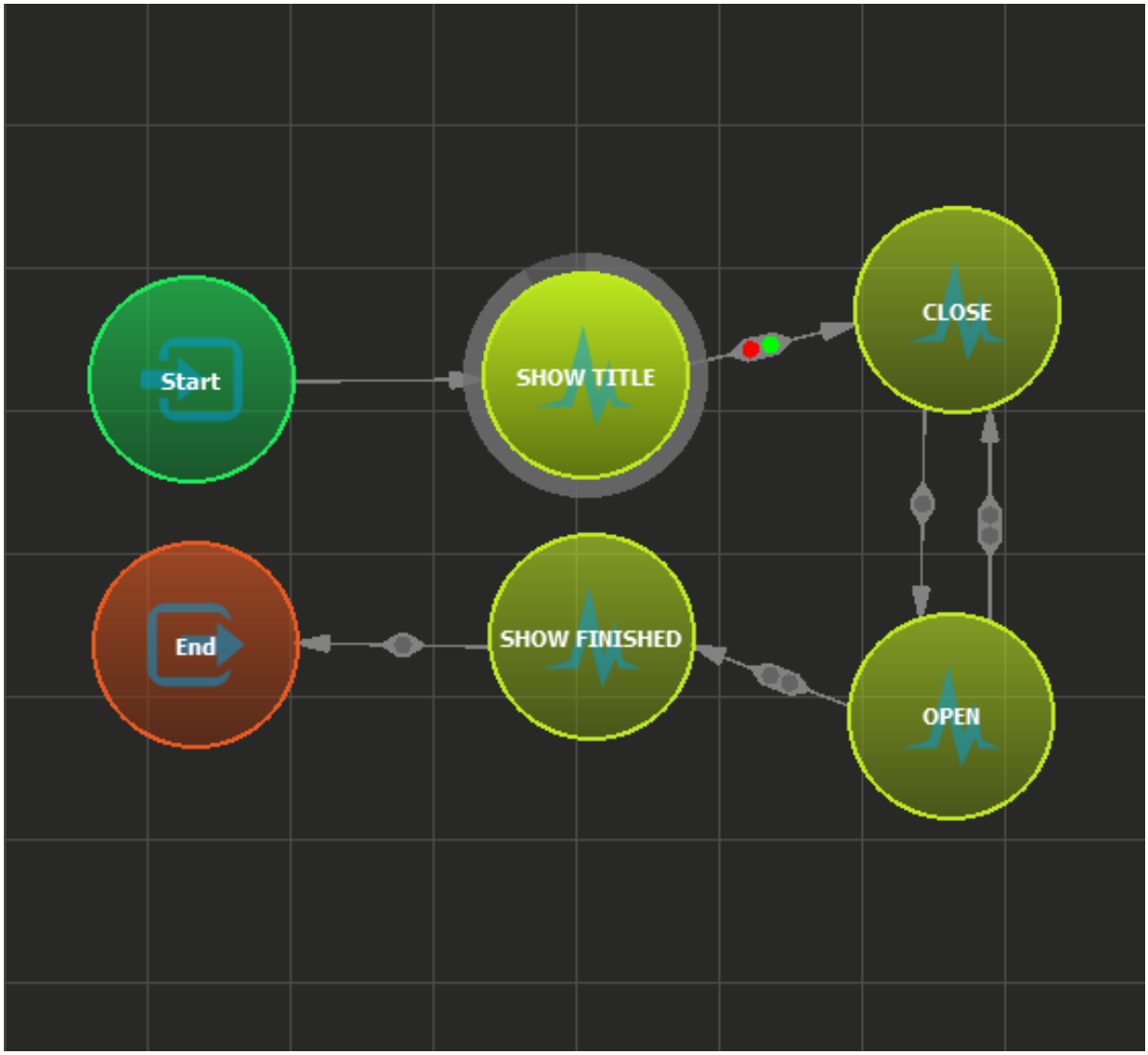


**Figure 4.5:** State-Machine of Experiment 1

### 4.3.7 Visualization Description

The performance of a participant can be visualized comprehensively and analyzed by a graph as presented in Figure 4.6 on the next page. Such visualization for *Experiment 1* can be reviewed in the appendix for all subjects (see A.1).

**Axes**:

All axes are linear. The x-axis is the time line, this is equal for all graphs and hence only shown once at the bottom of a visualization. The right (orange) y-axes always refer to the reference value. The left y-axes vary from graph to graph and refer to the measured values.

**Curves**:

The *orange* curve is the reference value for the closeness state of the eyes in all graphs. It is set to *1* whenever the subject is commanded to close the eyes and it is set back to *0* whenever the subject is commanded to open the eyes again. This reference value is plotted once for each subgraph. As soon as the reference value has changed, the different detectors should trigger after a subject dependent reaction time.

In the first graph, the *blue* curve shows the binary classification for the closeness state of the eyes and the *green* curve is the regression value for the closeness state of the eyes. Both are extracted using *OpenFace*++ (AU45). The binary classifier (blue) should have a *0* for closed eyes and a *1* for opened eyes. The regression value (green) has a high value for closed eyes and a small one for opened eyes (see 4.3.9 for more details).

The others graphs below the first one all visualize the data retrieved from the *Emotiv EPOC*, such as the binary eye-blink classifier. Note that the eye-blink detector is just an *AND* combination of the closeness state of left and right eye, so below the graph for eye-blink, the left and right detections are dedicatedly plotted. The last two graphs show the raw EEG data that was captured at the frontal *AF3* and *AF4* electrodes (see 2.3.1). For most subjects, some kind of pattern is clearly visible for the eye-activity.

**Remarks**:

The period before the first close stimulus must be considered as a naturally opened or rather undefined eye state. Subjects were allowed to perform final eye-blinks or other preparations for the experiment in this interval and were not yet explicitly commanded to keep the eyes opened.

### 4.3.8 Data Visualization

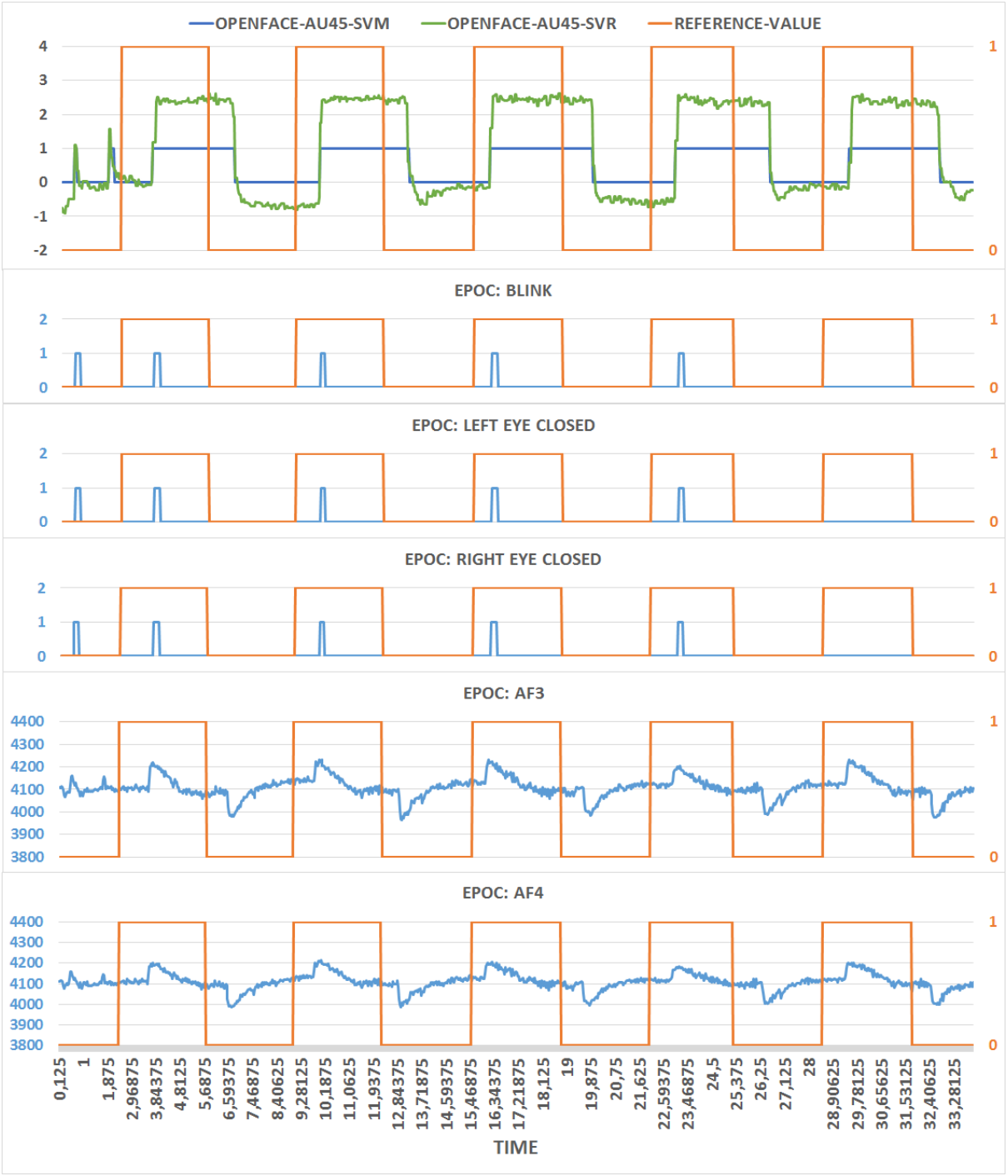


**Figure 4.6:** Performance of Subject 02

### 4.3.9 The AU45 Regression

The AU45 regression value of OpenFace is maximal and minimal for fully closed and opened eyes. However, the exact values of the maximum and minimum depend on subject and environment. Observations suggest the assumption that glasses, face orientation, distance and illumination conditions can all influence the value.

Table 4.1 below presents the AU45 regression values which were captured in *Experiment 1* under rather perfect conditions. For each subject the global minimum, maximum and the difference between them is listed. Additionally, the average value of the periods with closed eyes, the average value of the periods with opened eyes and the difference between both is shown. The lower part of the table provides statistics calculated over all subjects. This includes the minimum of all minimums, the maximum of all maximums and so on.

| | OPENFACE AU45 SVR | | | | | |
|---|---|---|---|---|---|---|
| **Subject** | **MIN** | **MAX** | **DELTA** | **CLOSE**[2] | **OPEN**[2] | **DELTA** |
| 01[1] | +0.165 | +2.739 | +2.573 | +2.527 | +0.425 | +2.101 |
| 02 | -0.904 | +2.619 | +3.524 | +2.407 | -0.382 | +2.789 |
| 03 | -0.872 | +2.888 | +3.761 | +2.183 | -0.174 | +2.358 |
| 04 | -1.227 | +3.128 | +4.356 | +2.612 | -0.971 | +3.584 |
| 05 | -0.467 | +2.959 | +3.426 | +2.570 | -0.296 | +2.867 |
| 06 | -1.018 | +3.573 | +4.592 | +3.234 | -0.776 | +4.010 |
| 07[1] | -0.346 | +2.576 | +2.923 | +1.461 | -0.085 | +1.547 |
| 08 | -0.740 | +3.042 | +3.782 | +2.501 | -0.385 | +2.887 |
| **MIN** | -1.227 | +2.576 | +2.573 | +1.461 | -0.971 | +1.547 |
| **MAX** | +0.165 | +3.573 | +4.592 | +3.234 | +0.425 | +4.010 |
| **MED** | -0.806 | +2.923 | +3.642 | +2.514 | -0.339 | +2.828 |
| **AVG** | -0.676 | +2.941 | +3.617 | +2.437 | -0.319 | +2.768 |
| **SD** | +0.443 | +0.321 | +0.671 | +0.494 | +0.426 | +0.787 |
| **MD** | +0.345 | +0.234 | +0.505 | +0.314 | +0.298 | +0.574 |

**Table 4.1:** OpenFace AU45 SVR Values

[1] Glasses
[2] Average of all five periods

The undefined boundaries of the AU45 regression value are not suitable with regard to implicating the current closeness state of the eyes directly. Without such boundaries, a single regression value can theoretically mean anything. But once the boundaries are defined, the value gets a meaning and can be optionally translated into a normalized, more user-friendly interval, such as [0..1].

Figure 4.7 below visualizes the most important data from previous table. For each subject, the average (blue) and maximum (yellow) AU45 regression values for closed eyes and the average (orange) and minimum (gray) AU45 regression values for opened eyes are shown.

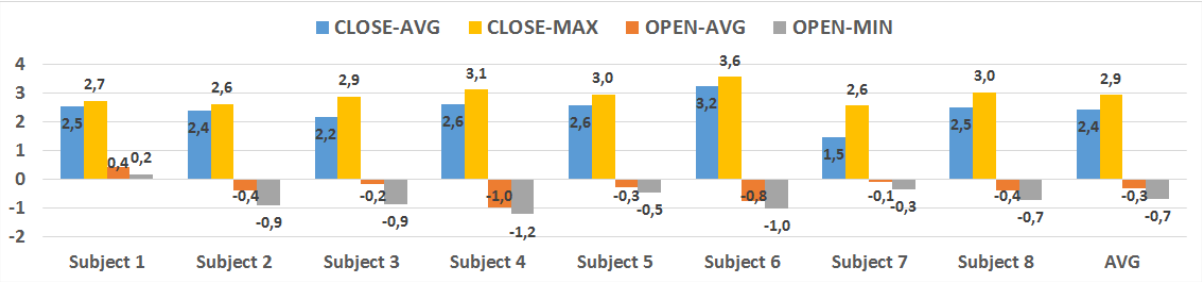


**Figure 4.7:** AU45 values of subjects

**Closed-Eyes Periods**: The smallest average AU45 regression value was measured for subject 07 (*1.461*). Subject 06 showed the highest average value (*3.234*). The average of all closed eye periods of all subjects is *2.437*.

**Opened-Eyes Periods**: The highest average AU45 regression value was measured for subject 01 (*0.425*). Subject 04 showed the lowest average value (*-0.971*). The average of all opened eye periods of all subjects is *-0.319*.

**Glasses**: It's notable that both subjects with glasses have the highest two average values (*0.425* and *-0.085*) for opened eyes. Subject 07 additionally also has the lowest average (*1.461*) and subject 01 the fourth lowest average (*2.527*) for closed eyes. This results in both having the smallest interval size and hence also smallest measurable difference between closed and opened eyes states.

**Conclusion**: Classifying regression values above 2.0 as likely closed eyes, values below 0.0 as rather opened eyes and anything in between as half-opened would yield an acceptable approximation for these subjects. However, it's better to implement a mapping which is based on the so far seen maximum and minimum values for the current environment parameters, such as distance, orientation and subject. To do so, the maximum and minimum must be tracked per subject and then used to map a detected value into a normalized [0..1] interval.

### 4.3.10 Metric 1 - Detections

Figure 4.8 shows a perfect transition from opened to closed eyes.

The orange reference value changes first. Then, after a delay dependent on the subject's response time, the three different detectors trigger.

Five of this feature should be visible in a perfect run. A correct detection is defined as a transition from *0* to *1* in the binary classifiers or a peek to *2.0* and above in the regression.

Table 4.2 shows the performance of each detection type with regard to this feature. Incorrect detections are investigated next.

**Figure 4.8:** Example

| | OpenFace++ | | Emotiv EPOC | |
|---|---|---|---|---|
| **Subject** | **AU45 SVM** | **AU45 SVR** | **EYE-BLINK** | **MIN-ONE-CLOSED** |
| 01[1] | 0/5 | 5/5 | 5/5 | 5/5 |
| 02 | 5/5 | 5/5 | 4/5 | 4/5 |
| 03 | 2/5 | 5/5 | 0/5 | 5/5 |
| 04 | 5/5 | 5/5 | 0/5 | 5/5 |
| 05 | 5/5 | 5/5 | 4/5 | 5/5 |
| 06 | 5/5 | 5/5 | 0/5 | 0/5 |
| 07[1] | 5/5 | 5/5 | 5/5 | 5/5 |
| 08 | 3/5 | 5/5 | 5/5 | 5/5 |
| **Detections** | 31/40 | 40/40 | 23/40 | 34/40 |
| **Rate** | 78% | 100% | 58% | 85% |

**Table 4.2:** Results of Metric 1

[1] Glasses

### 4.3.11 Metric 2 - Errors

Figure 4.9 shows a period of opened eyes with incorrect detections in it.

The blue curve (*OpenFace++ AU45 SVM*) shows at least two incorrectly detected states of closed eyes. The lower graph (*Emotiv EPOC Blink*) triggers unexpectedly for opening the eyes, which occurred for only 3 of 40 transitions from closed to opened and hence is categorized as an error.

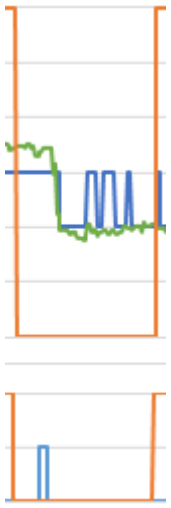

**Figure 4.9:** Example

For each detector type, the periods of closed and opened eyes which contain at least one *detection error* were counted and compared in Table 4.3. There are five periods with closed eyes and five with opened, hence a maximum of ten failures can be collected.

For blue, one or more switch to the wrong classification within an interval counts as *detection error*. For green, violating the defined thresholds within an interval counts as *detection error* and for the BCI, any detection except for when closing the eyes is counted as *detection error*.

| | **OpenFace++** | | **Emotiv EPOC** | |
|---|---|---|---|---|
| **Subject** | **AU45 SVM** | **AU45 SVR** | **EYE-BLINK** | **MIN-ONE-CLOSED** |
| 01[1] | 5/10 | 0/10 | 0/10 | 0/10 |
| 02 | 0/10 | 0/10 | 0/10 | 0/10 |
| 03 | 5/10 | 0/10 | 0/10 | 0/10 |
| 04 | 0/10 | 0/10 | 0/10 | 0/10 |
| 05 | 1/10 | 0/10 | 0/10 | 2/10 |
| 06 | 0/10 | 0/10 | 0/10 | 0/10 |
| 07[1] | 3/10 | 0/10 | 2/10 | 2/10 |
| 08 | 4/10 | 0/10 | 2/10 | 2/10 |
| **Errors** | 18/80 | 0/80 | 4/80 | 6/80 |
| **Rate** | 23% | 0% | 5% | 8% |

**Table 4.3:** Results of Metric 2

[1] Glasses

### 4.3.12 Discussion

The most significant insight from *Experiment 1* is the fact that the *Emotiv EPOC* does not detect an actual closeness state of the eyes at all. Instead, only the eye-blink muscle activity is detected and propagated over a short period of time. This puts the *Emotiv EPOC* at a basic disadvantage compared to the capabilities of its rival, the facial data extraction by *OpenFace++*, which is able to detect the eye closeness state at all time.

Metric 1 and 2 show that:

- The AU45 regression of *OpenFace++* performs best, even when detecting and measuring only the transitions from opened to closed eyes, which both detection systems are capable of. The AU45 regression shows the highest detection rate (100%) and the smallest error rate (0%).
- The binary AU45 classifier of *OpenFace++* does not perform bad in Metric 1 (78%). But it fails completely for subject 01 with glasses. Furthermore, it also shows the highest error rate in Metric 2 (23%) and hence seems not very reliable.
- The *Emotiv EPOC* performs good in Metric 1 (85%), if the requirements are reduced so that detecting only one of the two closing eyes is sufficient. If both are required, the detection rate drops to 58%. However, the error rate of the *Emotiv EPOC* for both cases is good (8% and 5%).

In summary, it can be stated that the AU45 regression of *OpenFace++* outperforms all other candidates. But to be fair, it must be pointed out, that *OpenFace++* was operating at almost perfect conditions in the experiment with regard to illumination, face size and orientation. A BCI driven approach starts to gain advantages when these conditions start to get worse. However, BCI devices are quite unhandy to handle and hence come with their own big disadvantage that can also cause them to function incorrectly.

Finally, the hypothesis (4.3.1) turns out to be correct. Image processing based eye-blink detection has shown a reliable and equally or even better performance than a comparable, but much more expensive BCI driven approach.

## 4.4 Experiment 2

The second experiment evaluates the position and orientation recognition capabilities of *OpenFace++*. *OpenFace++* and the original *OpenFace* are able to track the current 3D position and orientation of the face. The recognized motions are compared to the data retrievable from alternative sensors, such as a BCI-integrated gyrometer.

### 4.4.1 Hypothesis

Image processing based face orientation detection performs reliably and equally or even better than a gyrometer embedded in a head mounted BCI.

### 4.4.2 Task

The participant is requested to face a specific direction on command. The possible directions are *Left* (L), *Right* (R) and *Center* (C). L and R are specified by markers in the room, but the absolute position is less important here. The stimuli to change the facing direction occur in intervals of 3 seconds with 4 iterations on the combination LCRC and a centered starting position and orientation. One trial run is granted to get familiar with the task.

### 4.4.3 Independent Variable

The first controlled variable in the experiment is the detection system, namely the *OpenFace++* library or the gyrometer embedded in the *Emotiv EPOC* BCI. The second controlled variable is the direction that the subject is commanded to face towards.

### 4.4.4 Dependent Variable

The measured variable is the performance of the different detection approaches.

### 4.4.5 Neuromore Classifier

The classifier of Experiment 2 uses the *Emotiv EPOC* device, three parameters and nine extracted facial data values, such as position and orientation, provided by *OpenFace++*. The reference value is parameter *MOVE*. The raw EEG data is not captured in this experiment, only the *Yaw* and *Pitch* values of the gyroscope are recorded for comparison. All data is written to CSV. Operates at 32 Hz.

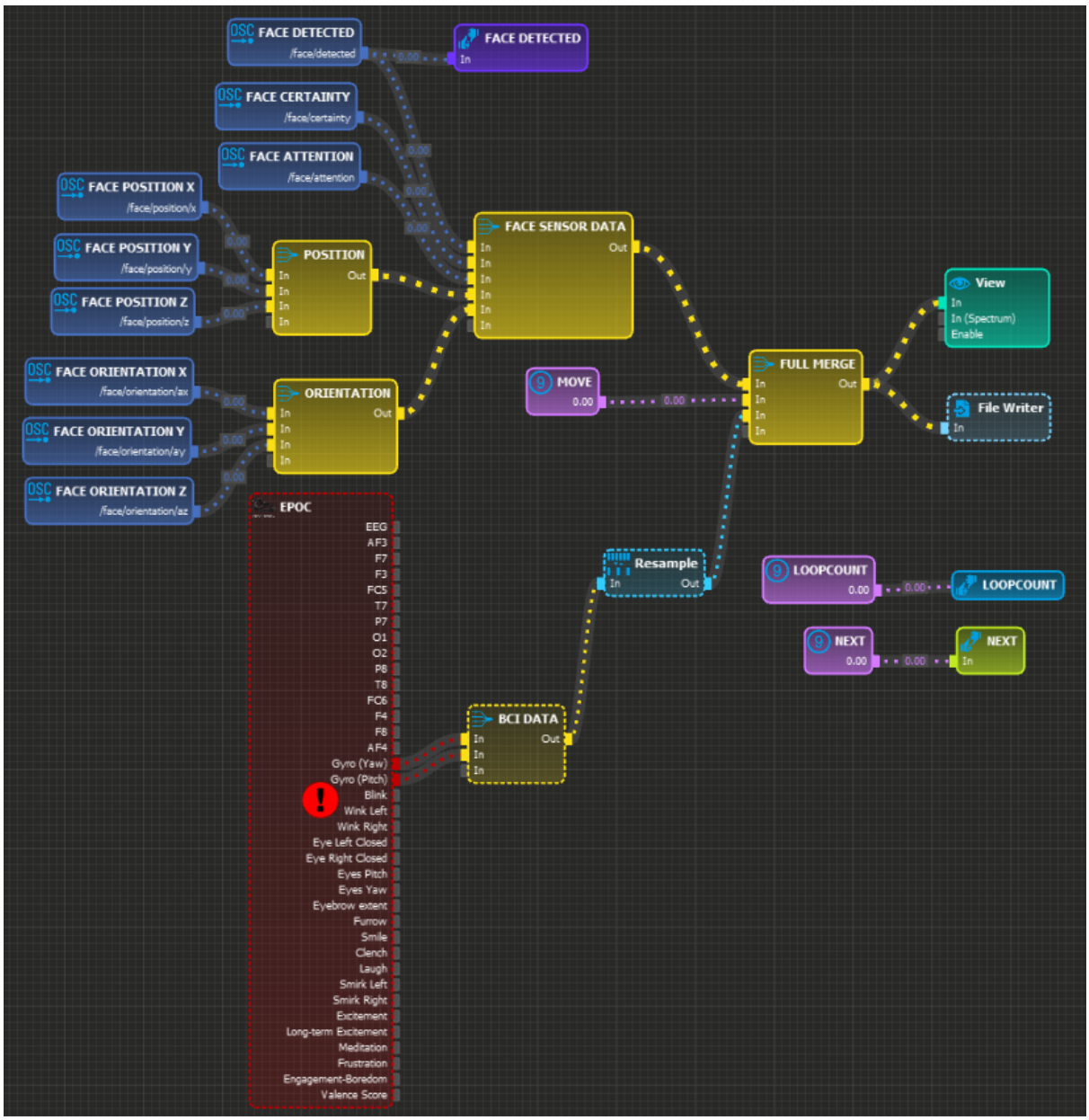


**Figure 4.10:** Classifier of Experiment 2

### 4.4.6 Neuromore State-Machine

The state-machine of Experiment 2 includes five internal states. Entering the *Left*, *Right* and *Center* states triggers the according audio and visual stimuli and adjusts the *MOVE* parameter in the classifier. The *NEXT* parameter is used to iterate the decision between *Left* and *Right*.


**Figure 4.11:** State-Machine of Experiment 2

### 4.4.7 Visualization Description

The performance of a participant can be visualized comprehensively and analyzed by a graph as presented in Figure 4.12 and 4.13 on the next page. Such visualization for *Experiment 2* can be reviewed in the appendix for all subjects (see A.2).

**Axes**:

All axes are linear. The x-axis is the time line, this is equal for both graphs in the visualization and hence only shown once at the bottom of each visualization.

In the first graph, the left (*black*) y-axis belongs to the *black* curve and *OpenFace++*. The right (*orange*) y-axis belongs to the *orange* curve and the head-mounted *Emotiv EPOC*. Note that both axes use different units. The left y-axis always shows the same *radian* interval. The right axis uses the unit of the *Emotiv EPOC* and its range can vary across visualizations. Plotting on two different linear y-axis so that both graphs create a visually best match is equal to the mathematical approach of finding a linear transformation that converts one unit into the other. If confused, just image both sensors would be thermometers, but one would be saving its values in *Celsius* and the other one in *Fahrenheit*.

In the second graph, the left (*blue*) y-axis is the detection certainty of *OpenFace++* and the right (*orange*) y-axis is used to draw the boolean face detected state. Both axis are identically across all visualizations.

**Curves**:

In the first graph, the *black* curve represents the recorded yaw-angles of *OpenFace*. The *orange* curve shows those of the *Emotiv Epoc*. Ideally, both curves should exactly match, since they show the same face orientations in same motions.

In the second graph, the *blue* curve is the detection certainty of *OpenFace*. If this value drops, it's likely that the orientation detection becomes inaccurate. The *orange* curve is the face detection state. If this value flips to *0*, then the face was completely lost and any captured yaw-angle will be *0* during that period.

**Remarks**:

The subjects performed the orientation changes of their faces with quite different velocity. Some turned their head fast, some slower. This results in some graphs having rather steep transitions while others are a lot smoother.

### 4.4.8 Data Visualization

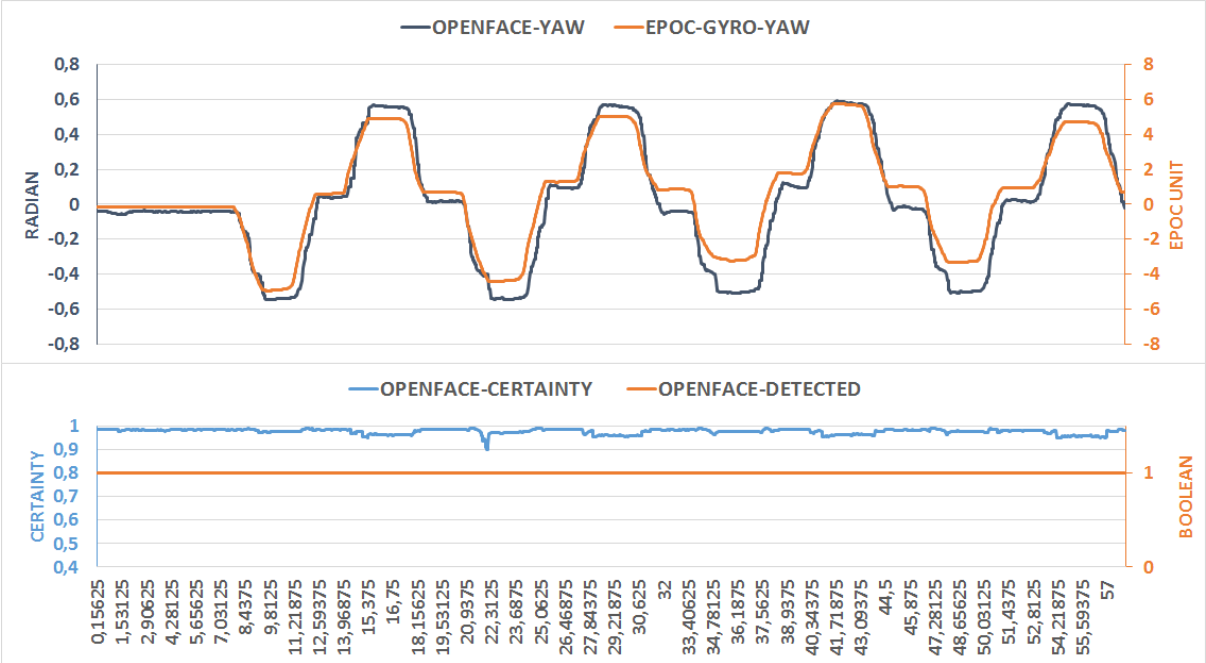


**Figure 4.12:** Performance of Subject 02

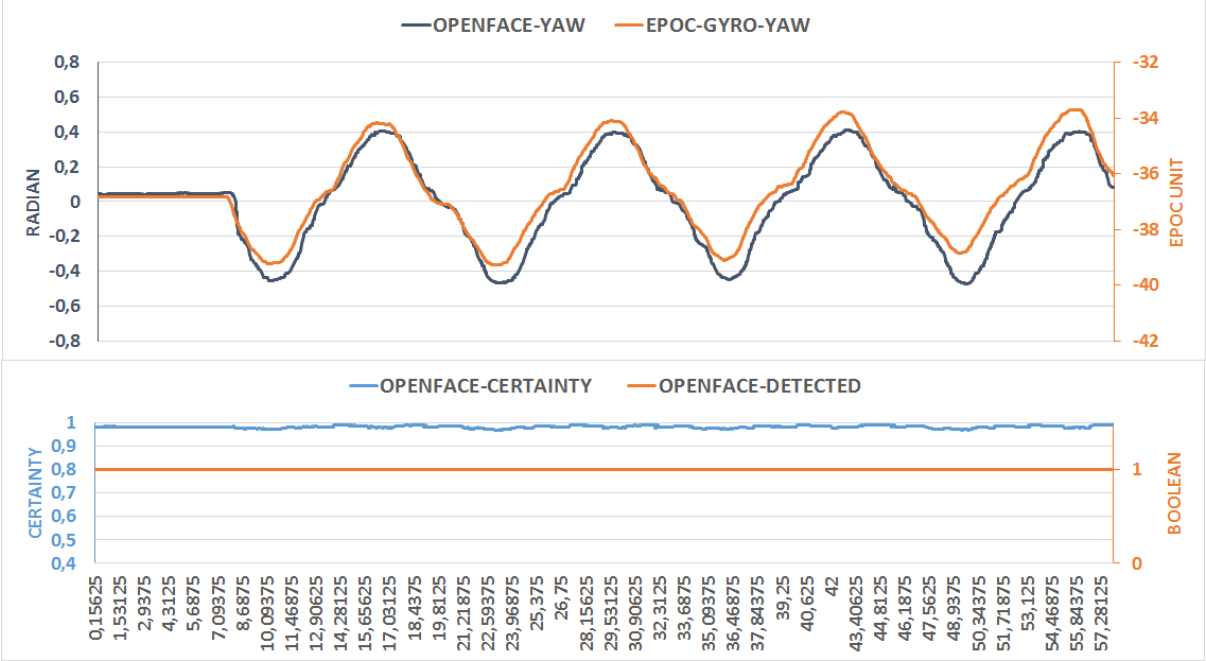


**Figure 4.13:** Performance of Subject 08

## 4.5 Experiment 3

The third experiment investigates the reliability of the new facial expression recognition in *OpenFace++* and investigates possible correlations between the results and additionally equipped bio-sensors such as a *HRM* or *GSR*.

### 4.5.1 Hypothesis

Image processing based facial expression detection in *OpenFace++* works reliably.

### 4.5.2 Task

The participant is asked to make six voluntary facial expressions on command. The example images of the different facial expressions from 3.5.2 are shown to the subject on the display. The images are supposed to serve as a hint. Subjects are told to imitate the facial expression, if in doubt how to perform it on command. Each of the facial expression must be held for 5 seconds. The subjects are requested to show a neutral face for 3 seconds between each of the six facial expressions. One trial run is granted to get familiar with the task.

### 4.5.3 Independent Variable

The only controlled variable in the experiment is the facial expression that subjects are requested to perform.

### 4.5.4 Dependent Variable

The measured variable is the performance of the system. The performance is measured with regard to detection rate and error rate.

### 4.5.5 Neuromore Classifier

The classifier of Experiment 3 uses the *Emotiv EPOC* device, two parameters and 14 extracted facial data values from *OpenFace++*. The reference value is parameter *EXPRESSION* which defines the currently expected facial expression. The raw EEG data is captured together with data from other bio-sensors, such as *HRM* and *GSR*. All data is written to CSV. Operates at 32 Hz.

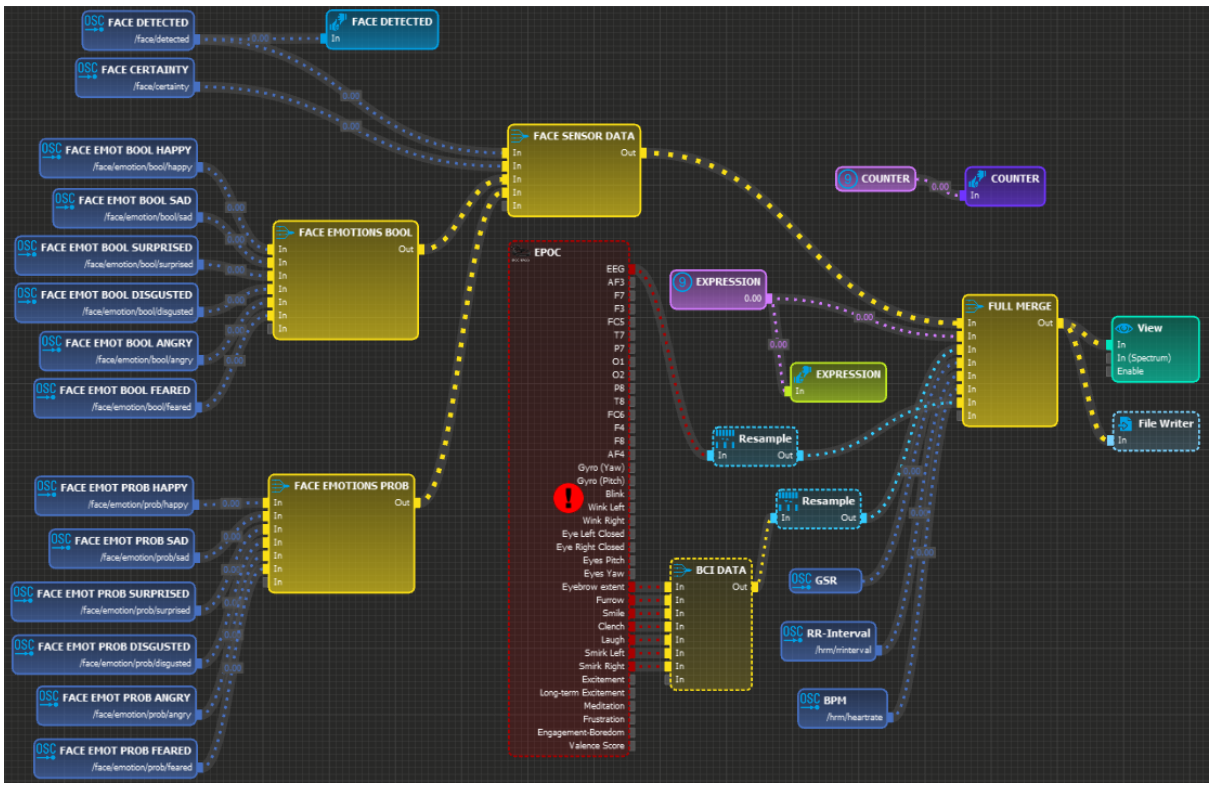


**Figure 4.14:** Classifier of Experiment 3

### 4.5.6 Neuromore State-Machine

The state-machine of Experiment 3 includes ten internal states. Entering the states named according to a facial expression triggers the visual stimuli that requests it from the subject by showing an example image and the name of the requested facial expression.

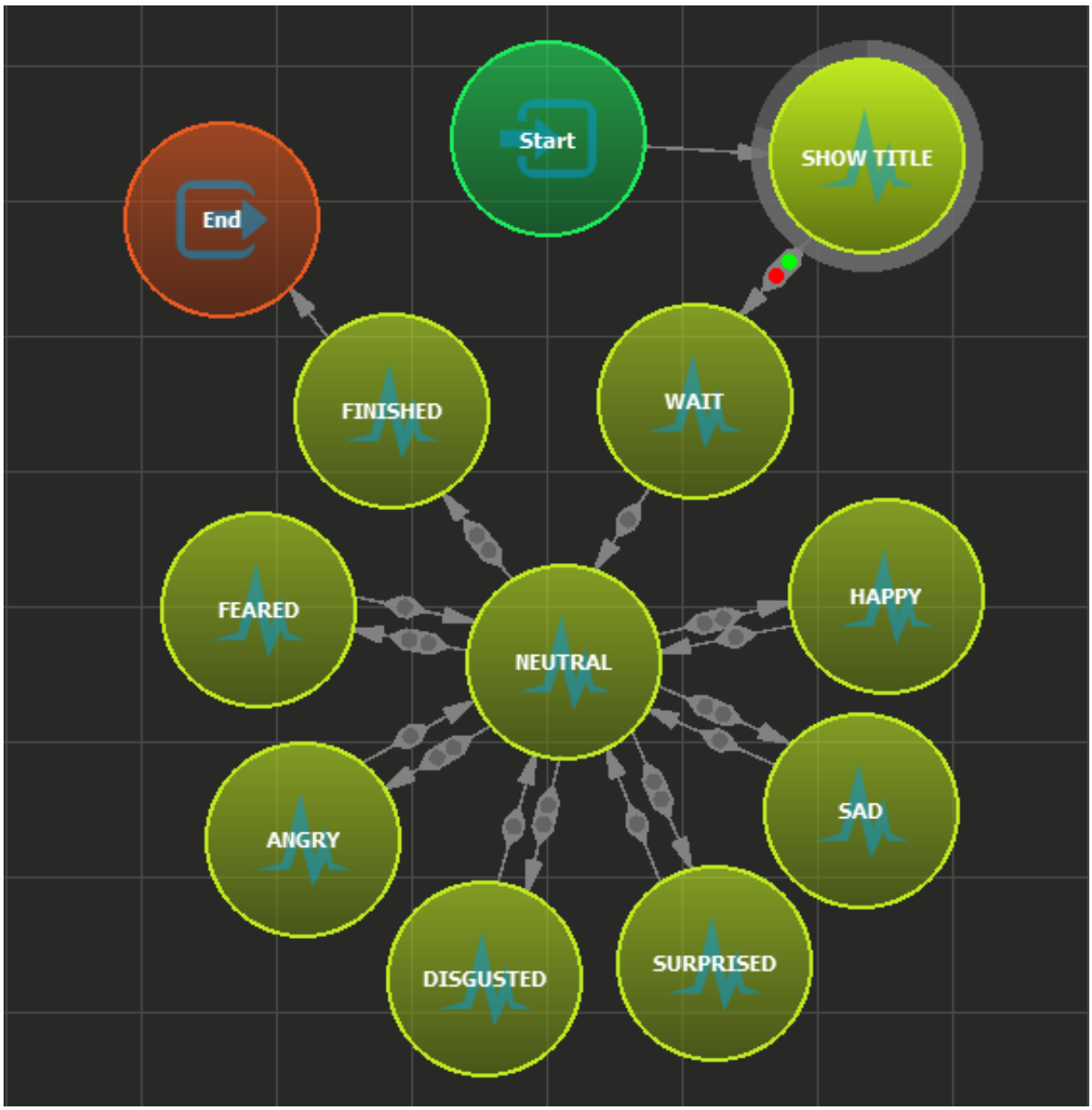


**Figure 4.15:** State-Machine of Experiment 3

### 4.5.7 Visualization Description

The performance of each participant can be visualized comprehensively and analyzed by a graph as presented in Figure 4.16 on the next page. Such visualization can be reviewed in the appendix for all subjects (see A.3).

**Axes**:

All axes are linear. The x-axis is the time line. It is equal for all graphs in the visualization and hence only shown once at the bottom of each visualization. The y-axis varies from graph to graph. For the first graph, it is the detected intensity of the different facial expressions which is retrieved from regressions. For the second to seventh graph, it is the logical expression classification whether that facial expression is active or not. For the next-to-last graph the y-axis is *microsiemens* measured by the GSR and for the last one it is *beats per minute* measured by the HRM.

**X-Intervals**:

The visualized and labeled facial expression intervals on the x-axis reflect the reference value. The subject was requested to perform the according facial expression during these periods.

**Curves**:

All graphs are colored according to the facial expression they represent. For instance, *brown* is *happy*. The *brown* curve ideally has a maximum period in the regression and a *1* in the happy-classifier for exactly the x-Interval which has *brown* borders and is marked as *Happy*. For all other periods the *brown* curve ideally has a minimum in the regression and a *0* in the classifier. All other facial expressions work accordingly.

### 4.5.8 Data Visualization

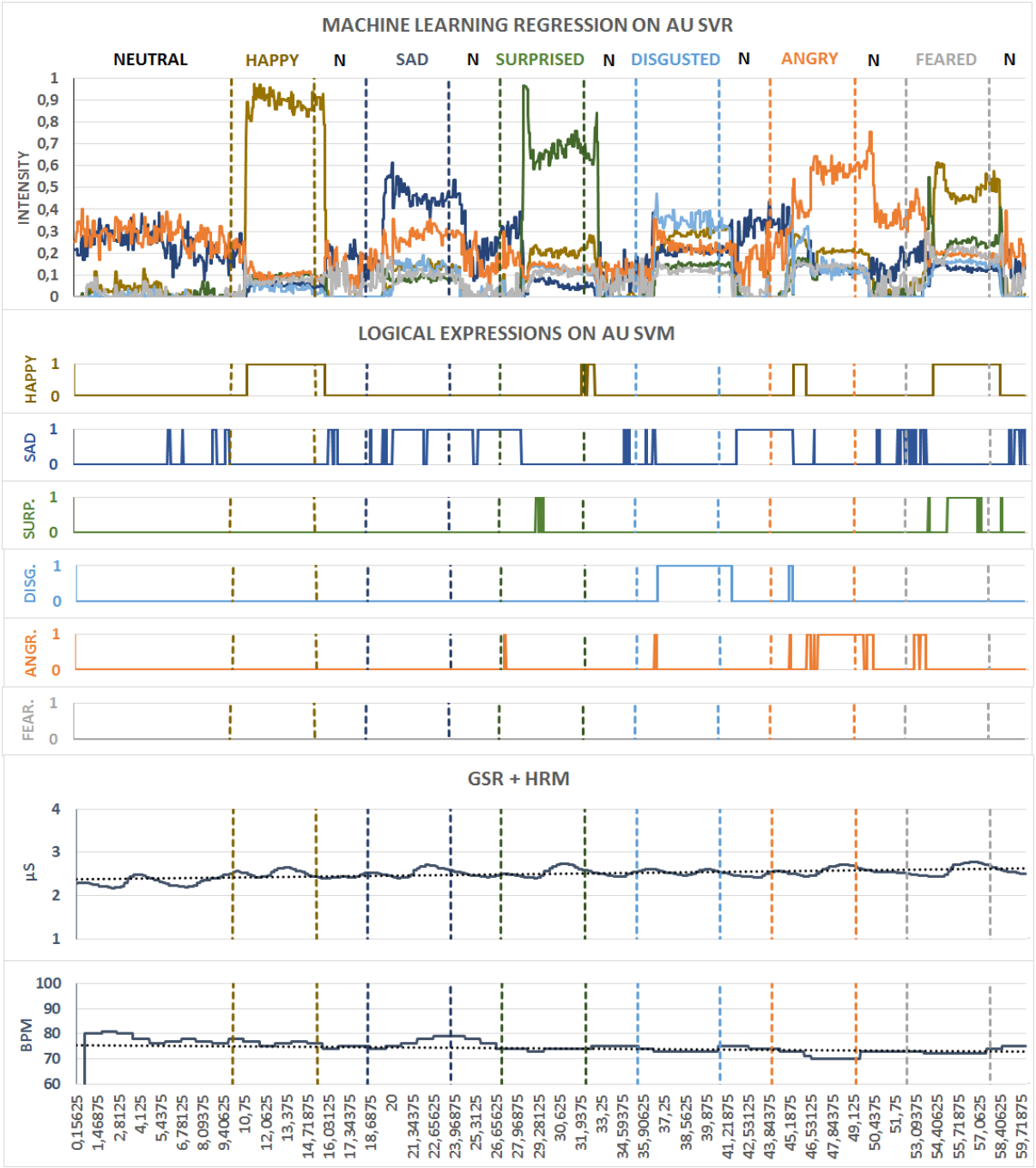


**Figure 4.16:** Performance of Subject 06

### 4.5.9 Metric 1 - Detections

This metric investigates the detection rate of the machine learning regression based facial expression detector (see 3.5.2). The performance is evaluated with regard to unambiguously detecting the according facial expression within the period in which it is actually requested.

To gain the maximum two points in this metric, the average regression value for the according facial expression period must be above a threshold of *0.5* and all other average regression values must be smaller by at least *0.3*.

To receive at least one point, the regression value for the requested facial expression must be above *0.3* and all others must be smaller by at least *0.1*.

For examples please see Figure 4.17 to the right. Table 4.4 below presents the results.

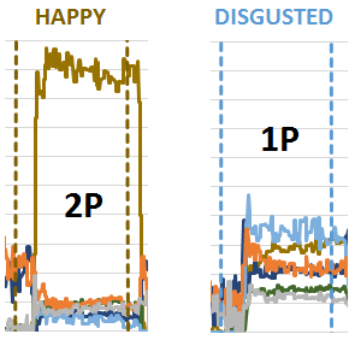


**Figure 4.17:** Example

| Subject | Regression HA | SA | SU | DI | AN | FE |
|---|---|---|---|---|---|---|
| 01 | 2/2 | 2/2 | 0/2 | 0/0[1] | 0/0[1] | 0/0[1] |
| 02 | 2/2 | 2/2 | 2/2 | 0/2 | 1/2 | 1/2 |
| 03 | 2/2 | 0/2 | 2/2 | 2/2 | 0/2 | 0/2 |
| 04 | 2/2 | 0/2 | 2/2 | 0/2 | 0/2 | 0/2 |
| 05 | 2/2 | 1/2 | 1/2 | 0/2 | 0/2 | 0/2 |
| 06 | 2/2 | 1/2 | 2/2 | 0/2 | 1/2 | 0/2 |
| 07 | 2/2 | 0/2 | 0/2 | 0/2 | 1/2 | 0/2 |
| 08 | 2/2 | 2/2 | 2/2 | 0/2 | 1/2 | 2/2 |
| **Score** | 16/16 | 8/16 | 11/16 | 2/14 | 4/14 | 3/14 |
| **Rate** | 100% | 50% | 69% | 14% | 28% | 21% |

**Table 4.4:** Metric 1 Results

[1]Not counted, subject was unintentionally smiling.

### 4.5.10 Metric 2 - Errors

This metric investigates the error rate of the machine learning facial expression regression.

An error in the regression is defined as a clear detection within periods that do not request that facial expression.

Given the overall of six requested facial expression periods and a regression which targets only one facial expression, each regression can make up to five errors in the five other periods.

Figure 4.18 shows the incorrect detection of *surprise* during a *fear* period. Table 4.5 presents the results.

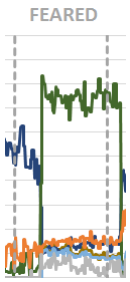


**Figure 4.18:** Example

| Subject | Regression Errors HA | SA | SU | DI | AN | FE |
|---|---|---|---|---|---|---|
| 01 | 0/2 | 0/2 | 0/2 | 0/3[1] | 0/3[1] | 0/3[1] |
| 02 | 0/5 | 0/5 | 0/5 | 0/5 | 1/5 | 0/5 |
| 03 | 0/5 | 0/5 | 1/5 | 1/5 | 0/5 | 0/5 |
| 04 | 0/5 | 0/5 | 1/5 | 0/5 | 0/5 | 0/5 |
| 05 | 0/5 | 1/5 | 0/5 | 0/5 | 0/5 | 0/5 |
| 06 | 1/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 |
| 07 | 0/5 | 0/5 | 0/5 | 0/5 | 2/5 | 0/5 |
| 08 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 |
| **Errors** | 1/37 | 1/37 | 2/37 | 1/38 | 3/38 | 0/38 |
| **Rate** | 3% | 3% | 5% | 3% | 8% | 0% |

**Table 4.5:** Metric 2 Results

[1]Not counted, subject was unintentionally smiling.

### 4.5.11 Metric 3 - Detections

This metric investigates the detection rate of the boolean classifiers which are implemented using logical expressions on detected AU (see 3.5.1).

To gain the maximum two points in this metric, the boolean classification must trigger throughout the whole period of the according facial expression without any loss of detection.

To receive one point, the classifier must trigger notably for most parts of the period.

For examples please see Figure 4.19. Table 4.6 presents the results.

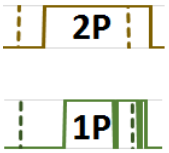


**Figure 4.19:** Example

| | Binary Classifiers | | | | | |
|---|---|---|---|---|---|---|
| **Subject** | **HA** | **SA** | **SU** | **DI** | **AN** | **FE** |
| 01 | 2/2 | 2/2 | 0/2 | 0/0[1] | 0/0[1] | 0/0[1] |
| 02 | 2/2 | 2/2 | 1/2 | 0/2 | 0/2 | 0/2 |
| 03 | 2/2 | 1/2 | 2/2 | 2/2 | 2/2 | 1/2 |
| 04 | 1/2 | 0/2 | 1/2 | 0/2 | 0/2 | 1/2 |
| 05 | 2/2 | 2/2 | 0/2 | 0/2 | 1/2 | 1/2 |
| 06 | 2/2 | 1/2 | 0/2 | 2/2 | 1/2 | 0/2 |
| 07 | 2/2 | 0/2 | 1/2 | 1/2 | 1/2 | 0/2 |
| 08 | 2/2 | 2/2 | 2/2 | 0/2 | 0/2 | 0/2 |
| **Score** | 15/16 | 10/16 | 7/16 | 5/14 | 5/14 | 3/14 |
| **Rate** | 94% | 63% | 44% | 36% | 36% | 21% |

**Table 4.6:** Metric 3 Results

[1]Not counted, subject was unintentionally smiling.

### 4.5.12 Metric 4 - Errors

This metric investigates the error rate of the boolean classifiers which are implemented using logical expressions on detected AU (see 3.5.1).

A boolean classifier error is defined as a clear detection within periods that do not request the according facial expression.

**Figure 4.20:** Example

Incorrect classifications which last only a rather short time are not counted with regard to better filtering techniques. Neutral periods are also not relevant.

Given the overall of six requested facial expression periods and a classifier which targets one facial expression, each classifier can make up to five errors in the five other periods.

Figure 4.20 shows the incorrect triggering of the *angry* classifier during a *disgusted* period. Table 4.7 presents the results.

| | **Binary Classifier Errors** | | | | | |
|---|---|---|---|---|---|---|
| **Subject** | **HA** | **SA** | **SU** | **DI** | **AN** | **FE** |
| 01 | 0/5 | 0/5 | 0/5 | 0/0[1] | 0/0[1] | 0/0[1] |
| 02 | 1/5 | 1/5 | 0/5 | 0/5 | 0/5 | 1/5 |
| 03 | 0/5 | 0/5 | 1/5 | 1/5 | 2/5 | 0/5 |
| 04 | 1/5 | 0/5 | 2/5 | 0/5 | 0/5 | 0/5 |
| 05 | 0/5 | 2/5 | 1/5 | 0/5 | 0/5 | 1/5 |
| 06 | 1/5 | 0/5 | 1/5 | 0/5 | 0/5 | 0/5 |
| 07 | 0/5 | 0/5 | 0/5 | 1/5 | 2/5 | 0/5 |
| 08 | 0/5 | 0/5 | 0/5 | 0/5 | 1/5 | 0/5 |
| **Errors** | 3/40 | 3/40 | 5/40 | 2/35 | 5/35 | 2/35 |
| **Rate** | 8% | 8% | 13% | 5% | 14% | 5% |

**Table 4.7:** Metric 4 Results

[1]Not counted, subject was unintentionally smiling.

### 4.5.13 Discussion

The plain results from the metrics are summarized first. Metric 1 and 2 evaluating the *regression* approach (see 3.5.2) show that:

- The *happy* regression detector performs best by far. It was able to detect the happy period of all subjects (100%) with only a single incorrect detection (3%) in periods of other facial expressions.
- The performance of the *surprised* and *sad* regression detectors is acceptable. *Surprise* achieved 69% of the possible score in detection with a low error rate of 5% and *sadness* scored 50% in detection with a low error rate of 3%.
- *Disgust*, *Anger* and *Fear* show the lowest performance. All of them achieve only less than 30% of the maximum score when it comes to detecting their own facial expression periods. *Disgust* holds the red lantern with only 14% of the possible detection score. Furthermore, *anger* has the highest error rate at 8%.

Metric 3 and 4 evaluating the *classification* approach (see 3.5.1) show comparable results:

- The *happy* classifier performs also best. It achieves a detection score of 94% with an error rate of only 8%.
- The *sad* and *surprised* classifiers perform averagely by correctly detecting 63% and 44% of their own intervals and incorrectly triggering for 8% and 13% of other facial expression periods.
- *Disgust*, *anger* and *fear* show the worst performance. They score 36%, 36% and 21% in detection with error rates of 5%, 15% and 5%. Like in the regression this also makes the *angry* classification the faultiest one.

It also needs to be pointed out that:

- Subject 01 clearly had to smile in the last three periods arguing he did not know how to make these requested facial expressions correctly. The affected three periods were not counted in the metrics.
- Several subjects stated that performing some of the requested voluntary facial expressions was not an easy task for them. As a result, the reference value in this experiment can not be assumed to be always correct. Also some subjects appeared to have a rather not neutral normal facial expression. For instance, subject 05 in particular seemed to have a high grade of *sadness* in his normal expression.

### Ambiguousness

Some facial expression regression detectors do not perform very well in Metric 1. One reason is that it's not trivial to show these facial expressions on command. But another one seems to be the great similarity between some of the facial expressions when it comes to the involved *Action Units* (see 2.2.4 and 3.5.3).

### Disgusted vs. Angry

Figure 4.21 to the right shows the *disgusted* period of subject 08 and the *angry* period of subject 03. Both received zero points in Metric 1, because the detected facial expression was not distinguishable enough between them.

This additionally also happened at least for subjects 02, 04 and 06 and suggests that merging these both detectors likely yields a notable improvement in the detection rate and detected intensity. At very least it suggests that *disgusted* and *angry* detections should not be interpreted as contradicting to each other.

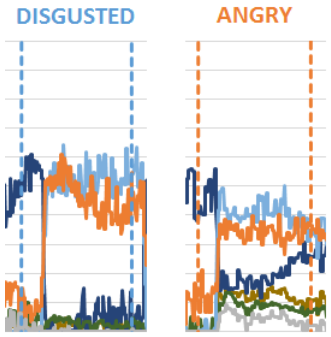


**Figure 4.21:** Example

### Surprised vs. Feared

Figure 4.22 to the right shows the *surprised* period of subject 05 and the *feared* period of subject 04. They received one and zero points in Metric 1 due to also blocking each other from gaining a more clear result and higher score.

This additionally also happened at least for subjects 02 and 03 and suggests that merging these two detectors also might yield a notable improvement. At very least it suggests that *surprised* and *feared* detections should not be interpreted as contradicting to each other.

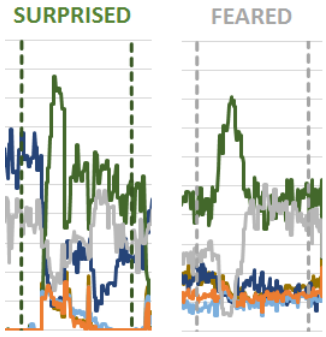


**Figure 4.22:** Example

Note that all machine learning regressions for facial expressions have been trained using an *One-vs all other* approach (see 3.5.2). In general this reduces the detected intensity for two regressions in case their samples can't be well enough separated. This is another good reason to think about merging classes, which can't be separated well enough.

### Age

The best performance was shown by subjects aged between 20 and 40. For both subjects with more than 60 years of age, the detection quality was rather average. It is also important to know that the machine learning regression (3.5.2) was only trained with samples from people aged between 20 and 40. It is possible that the regression works best for this group of people so far. For instance, older faces clearly show a higher base level of wrinkles than younger ones.

### Sensitivity

The results of subjects 03, 05 and 08 suggest that the sensitivity of the *sad* regression might be too high in general. Subject 07 also shows a high base level of *anger* and disgust, however this phenomena could not be found for other subjects.

### Classification vs. Regression

The classification performs surprisingly well, but it has a higher error rate in general. The regression yields the nice advantage of providing an actual intensity of the facial expression and hence is likely the better choice.

### Different Performances

As partially already pointed out in context of *ambiguousness* above, the facial expressions come with different difficulties in detecting them. For instance, the *happy* facial expression has a clear dedicated indicator (AU12). For other facial expressions the separation based on *Action Units* is not that trivial.

### GSR and HRM

The HRM data is missing for subjects 01 and 02 due to technical issues. For subject 05 the GSR and the HRM data is missing also. The captured datasets of the subjects show a wide range of different behaviors. The GSR ranges from very low values to very high values, to ones which barely change throughout the run and some which drastically change. The same is true for the HRM data. It is likely that the voluntary facial expressions and their rather short periods are not realistic enough to cause real measurable changes in these biosensors.

# 5 Conclusion and Future Work

Chapter 1 outlined several interesting applications for extracted facial data from ordinary webcams. Chapter 2 introduced the necessary background information of face analysis techniques and existing solutions. Chapter 3 explained the improved *OpenFace++* library, a major contribution of this thesis. Chapter 4 finally completed the work by presenting a user-study that evaluated the reliability of the *OpenFace++* library. *OpenFace++* is a cross-platform compatible, open source facial analysis framework based on the original *OpenFace*.

In conclusion, it can be stated that:

Facial data extraction can play an important role in understanding the user that operates a device. Several of the possible applications presented in the first chapter may soon become established successfully in commercial products or have been at least prototyped already.

Machine Learning approaches, such as *Support Vector Machines*, play an important role in classification and regression of complex data. Advanced frameworks exist that allow the application of machine learning approaches rather easily.

Extracting facial data has been a focus of computer scientists for a while now. Common sub-problems, such as initial face detection, have been tackled reliably and their solutions are reused widely across approaches.

There is a great gap between published research and commercial solutions for more sophisticated face analysis on the one side, and free, actually reproducible or open-source solutions on the other.

The original *OpenFace* framework has successfully started to close this gap between the functionality of commercial products and those actually available for free and open-source. *OpenFace* is a state-of-the-art facial data analysis framework written in C++11. It extracts 3D landmarks, face orientation, action units and eye-gaze.

The presented *OpenFace++* library aims to close this gap further by adding new features and cross-platform compatibility on top of the original *OpenFace*. For instance, the new *OpenFace++* additionally includes facial expression classification, attention estimation and comes with full Android support out-of-box.

Experiment 1 suggests that *OpenFace++* and the plain *OpenFace* can reliably detect the eye closeness state using the included AU45 regression detector. Furthermore, the facial data extraction approach outperformed the competing BCI eye-blink detection in the measured metrics. The BCI has complicated handling and only detects blink moments, not a persistent actual closeness state.

Experiment 2 showed that the orientation detection of *OpenFace++* is accurate with regard to the measured yaw motion in this experiment. The orientation data captured by the webcam very much matches the head-mounted gyrometer data for all subjects.

The performance of the facial expression detectors in experiment 3 was quite mixed. Detection of *happiness*, *sadness* and *surprise* worked rather well, while the recognition performance of *disgust*, *anger* and *fear* left much to be desired. A main reason for their bad performance is the ambiguousness of the according facial expressions. For instance, *disgust* and *anger* share several common *Action Units* which makes distinguishing them difficult, sometimes even for humans. Also several subjects mentioned difficulties in performing the facial expressions on command. No remarkable relations to the data captured by HRM and GSR devices were found.

## Future Work

*OpenFace++* still has a lot of potential for more improvements and extensions in the future. The recommended improvements for existing features are presented first. These mostly originate from the results of the user-study and they are:

- **Eye-Closeness State**: The AU45 regression output provides clear indication for the closeness state of the eyes. However, the subject and environment dependent minimum and maximum are not very user friendly with regard to easily retrieving a closeness state of the eyes. The value should internally be mapped into an additional output named *eyes-closed* which would be a value guaranteed to be within the interval [0..1] and easy expressible in percent.
- **Facial Expressions**: The results from experiment 3 suggest that the quality of the detection can be drastically increased for some facial expressions by merging their classes. The *anger* and *disgust* detectors should be considered for a merge into a combined detector. The same applies at least partially to the *surprise* and *fear* detectors. Mentioned facial expressions are hard to distinguish and the recommended merges will improve the performance drastically.
- **Facial Expression Classifiers**: The binary classifiers for facial expressions showed quite a lot of short detections across only a very few frames and these are often

incorrect. They can likely be suppressed using a more advanced filtering technique on a few of the last frames.

- **Android Support**: The performance requirements on mobile platforms are non negligible. *OpenFace++* showed about 10 FPS at high battery usage in tracking mode on a reference phone, but there is a lot of room for improvements on this topic. At very least the expensive *dlib* face detection should be replaced with the cheap *ViolaJones* detector. It is recommended to also evaluate if several of the internal and cached data which makes a large part of the memory requirements can be switched from double to single precision data types.

Additionally to these suggested improvements on existing functionality, there is a lot of room for more features. There is still a remaining gap to close compared to commercial state-of-the-art frameworks. The possible extensions include:

- **Age, Gender, Ethnicity Detection**: These are the missing main features compared to the data provided by some advanced commercial solutions. The possible applications range from age validation to surveillance. But especially for marketing applications this data is very interesting.
- **Eye Tracking**: The eye gaze vectors from facial data extraction are not very accurate yet. However, the accuracy should be sufficient to calculate at least an approximation of a focused area on a display in good environments. This would bring facial data extraction closer to the capabilities of professional infrared based eye-tracking installation and it would open up entirely new opportunities. However, this will require to manually provide information about the display, such as its size. An interesting study could be conducted to compare the accuracy of professional eye-tracking installations against ordinary webcams.
- **Multi Face Support**: *OpenFace++* and *OpenFace* support only a single face so far. Adding support for tracking and analyzing several faces simultaneously is clearly an interesting point. At least on more powerful systems this should be feasible and it will open up new possible applications, such as tracking a whole audience in the room. For most marketing applications this is also an essential requirement.

# A Study-Data

## A.1 Experiment 1

Visualizations of all data sets that were captured and investigated in *Experiment 1* are presented on the next pages.

For further details about that experiment please refer to section 4.3.

### A.1.1 Subject 01

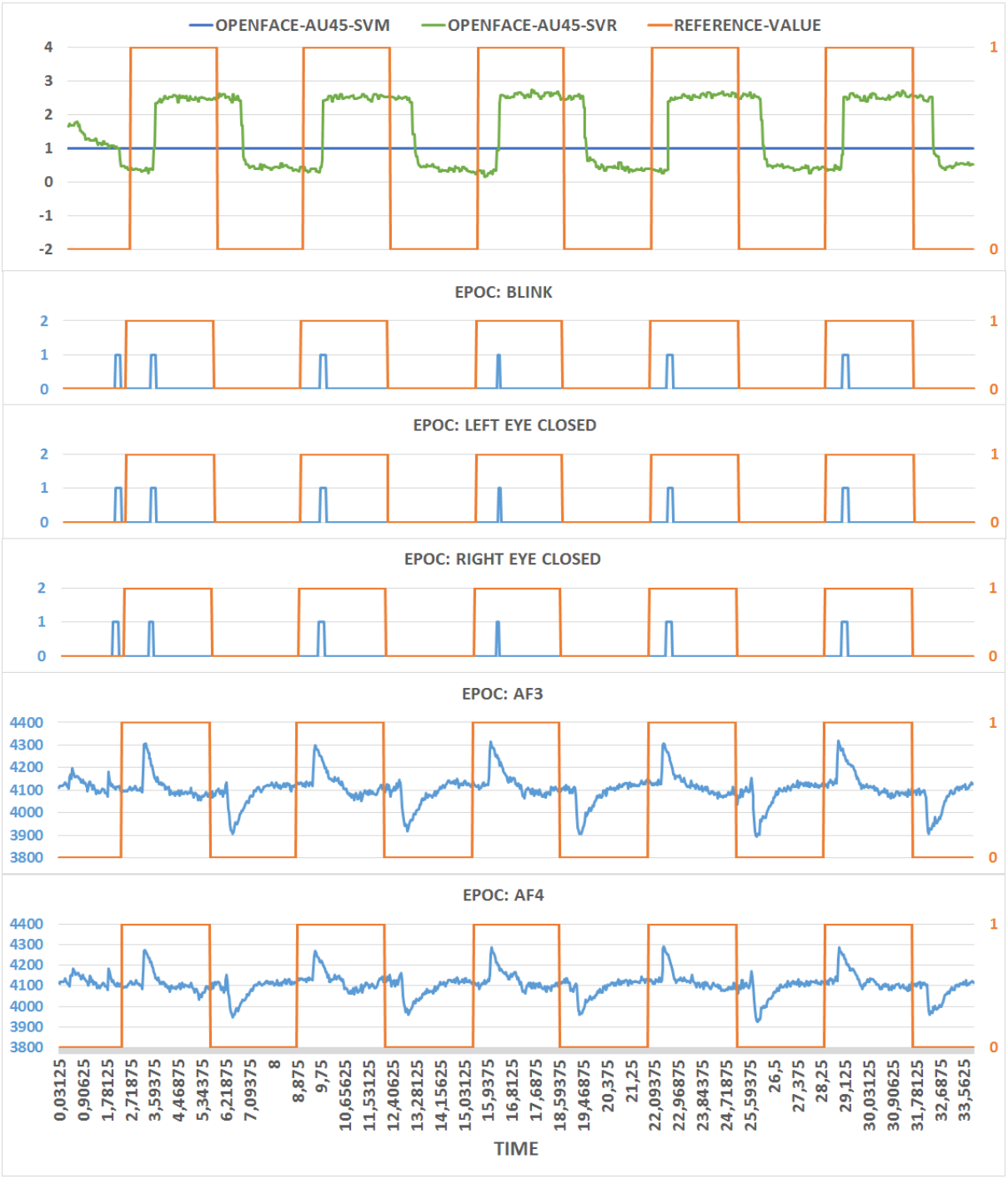


**Figure A.1:** Experiment 1 Data of Subject 01

### A.1.2 Subject 02

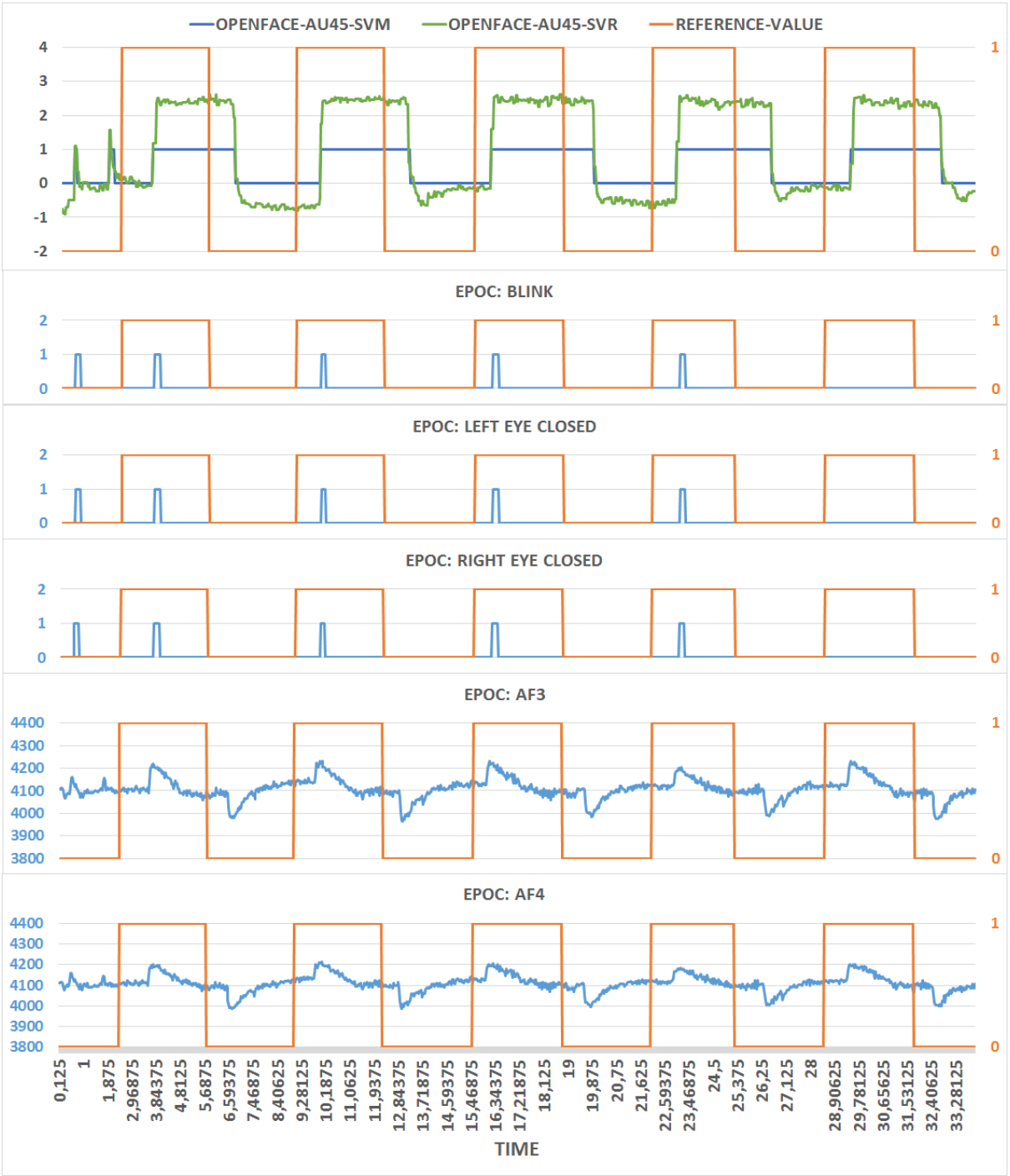


**Figure A.2:** Experiment 1 Data of Subject 02

### A.1.3 Subject 03

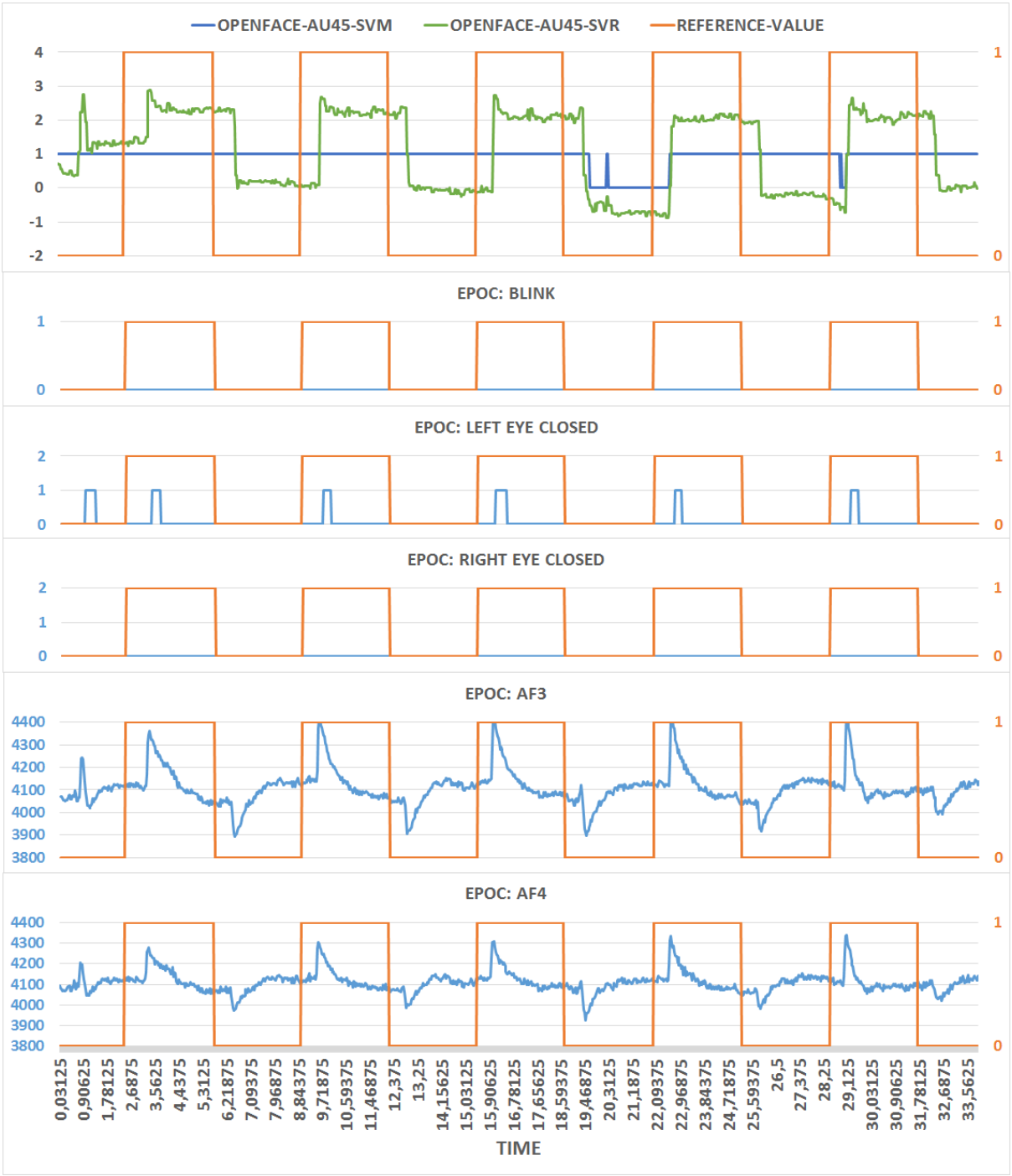


**Figure A.3:** Experiment 1 Data of Subject 03

### A.1.4 Subject 04

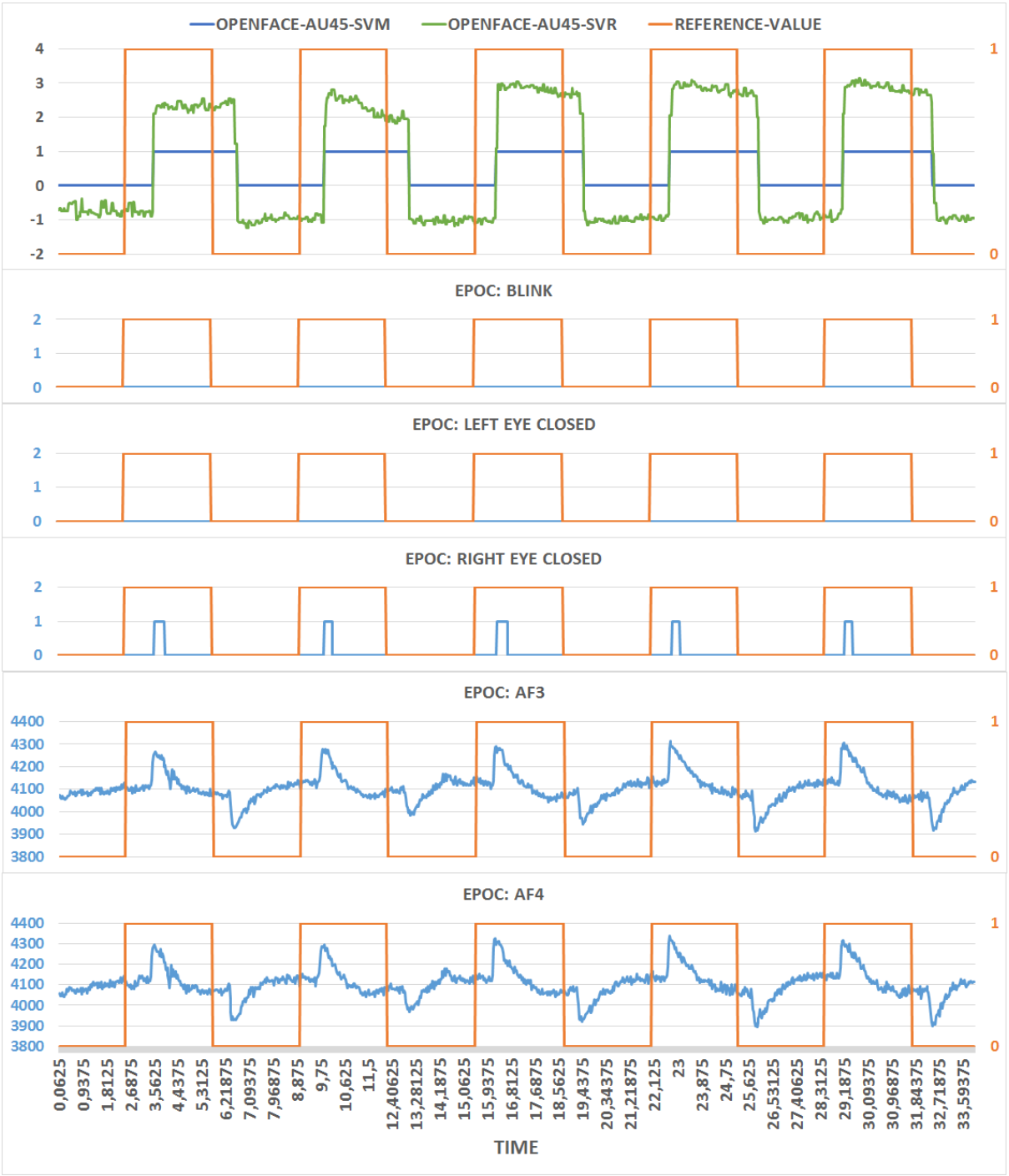


**Figure A.4:** Experiment 1 Data of Subject 04

### A.1.5 Subject 05

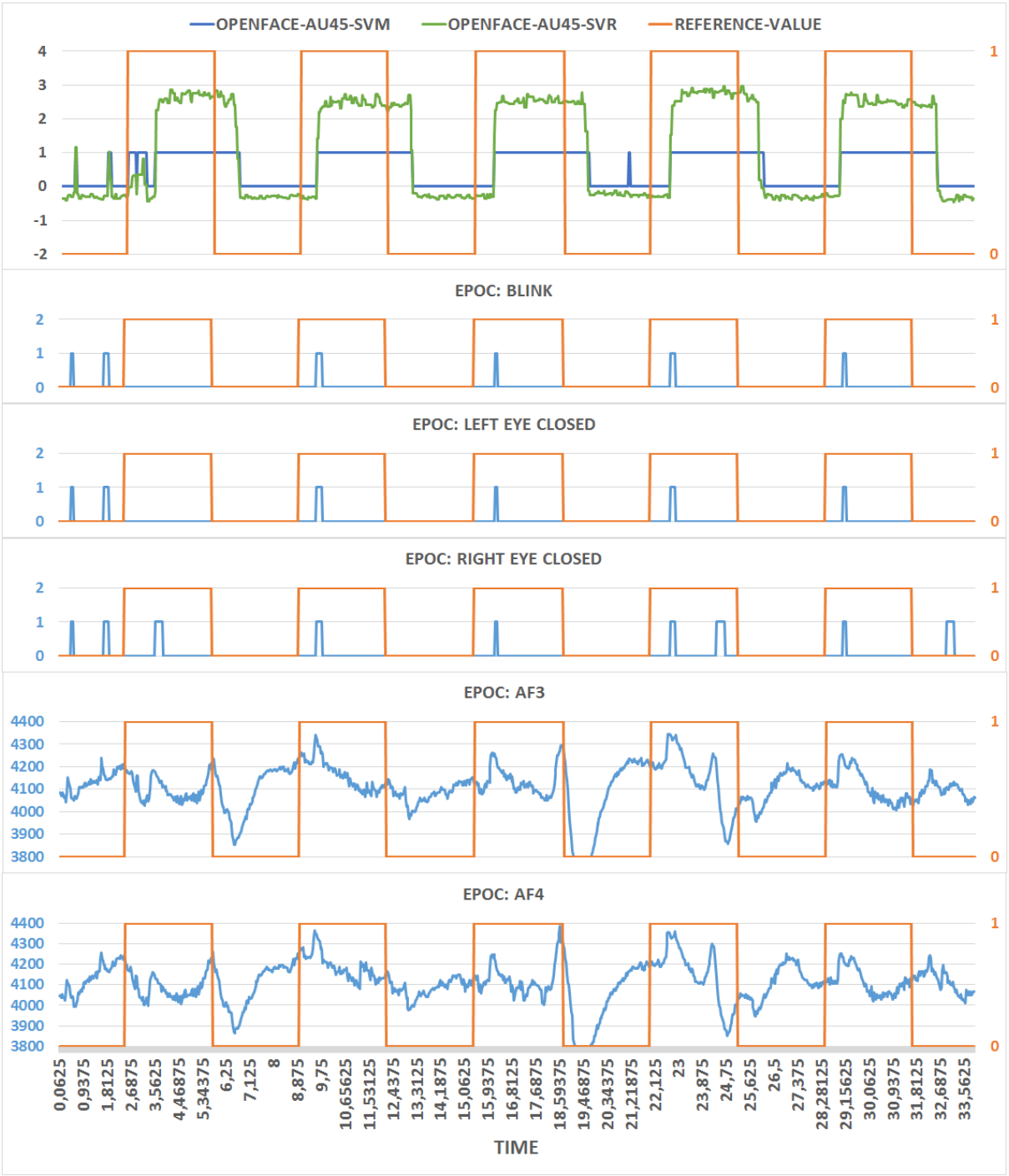


**Figure A.5:** Experiment 1 Data of Subject 05

### A.1.6 Subject 06

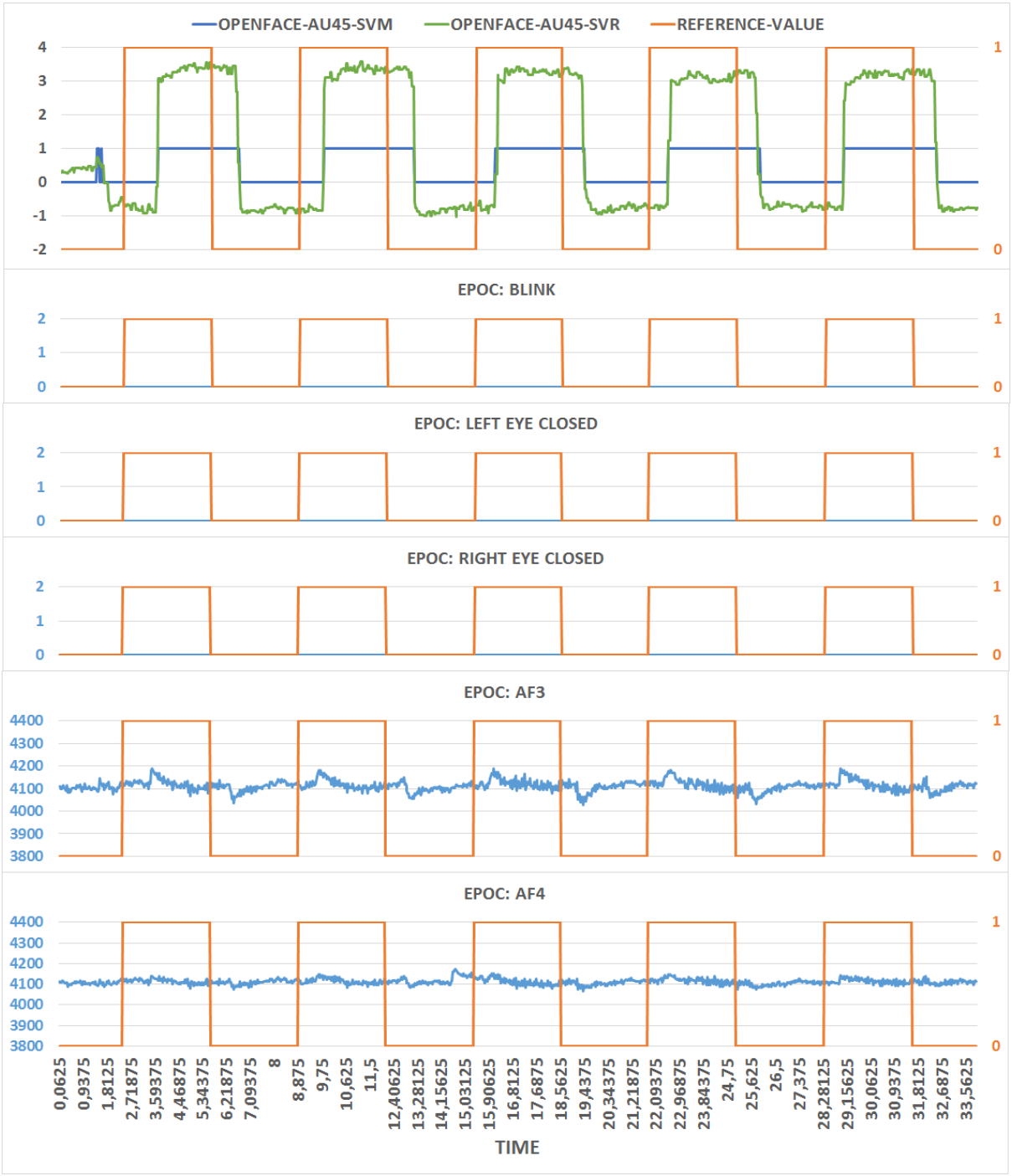


**Figure A.6:** Experiment 1 Data of Subject 06

### A.1.7 Subject 07

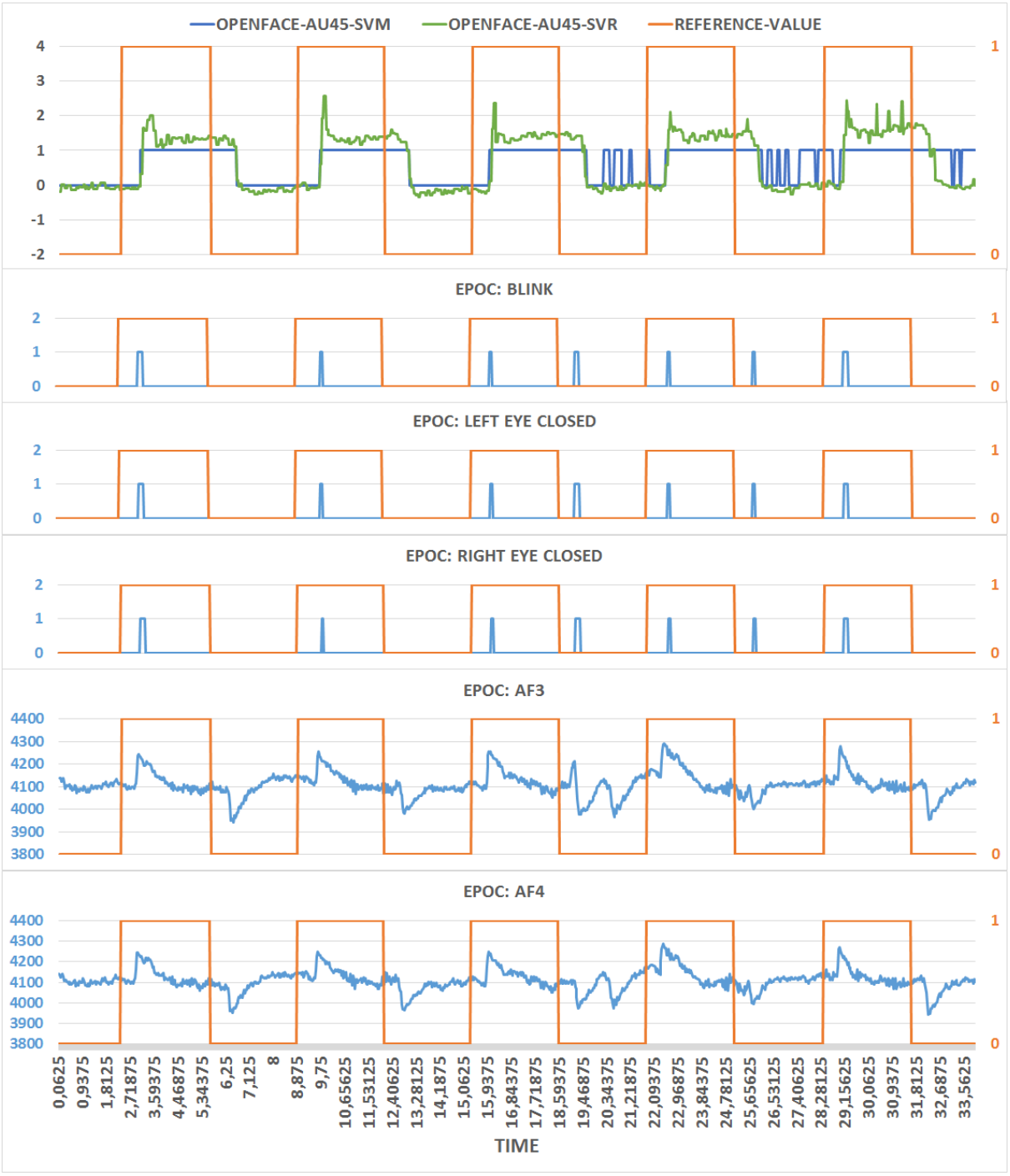


**Figure A.7:** Experiment 1 Data of Subject 07

### A.1.8 Subject 08

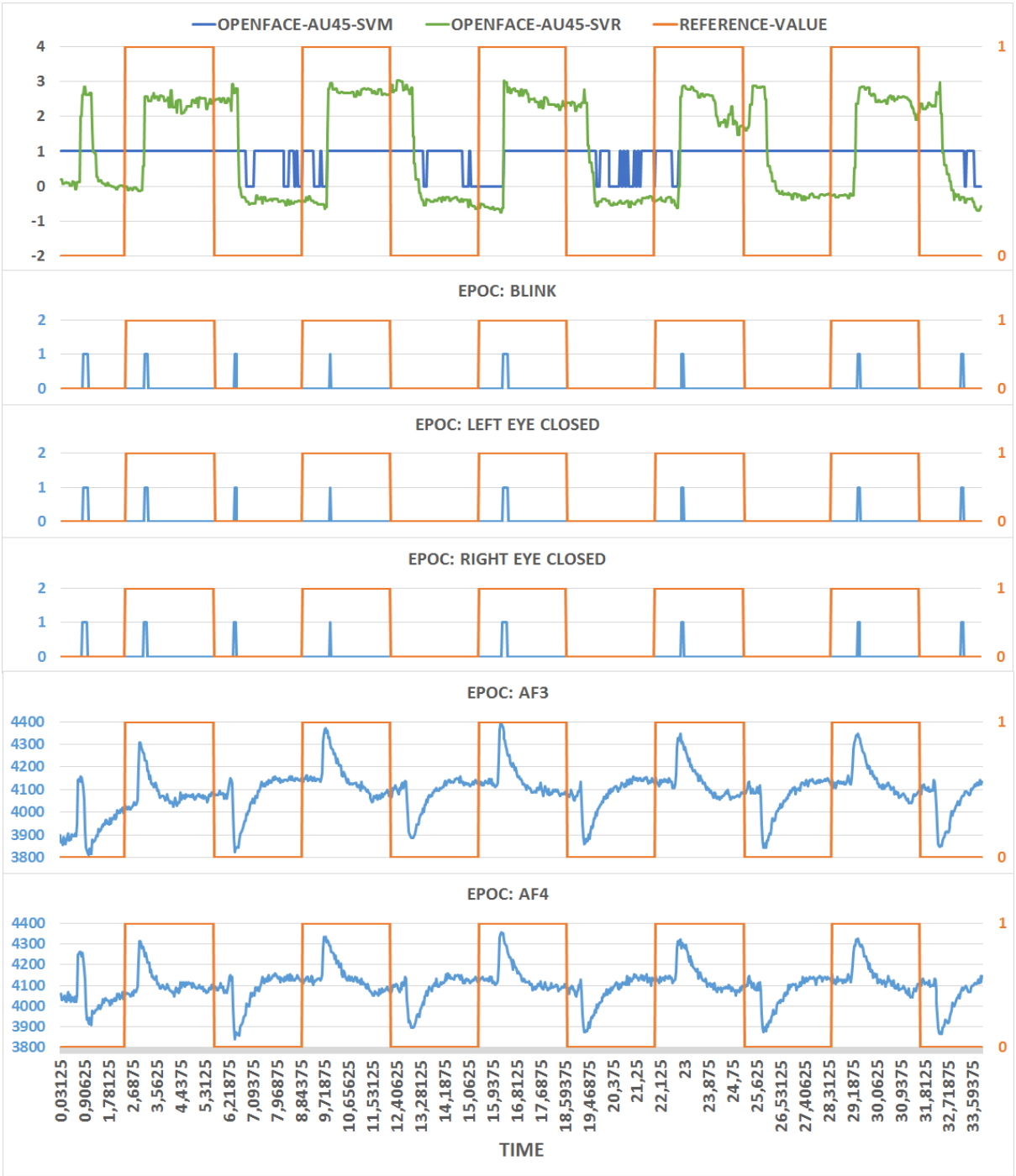


**Figure A.8:** Experiment 1 Data of Subject 08

## A.2 Experiment 2

Visualizations of all data sets that were captured and investigated in *Experiment 2* are presented on the next pages.

For further details about that experiment please refer to section 4.4.

## A.2.1 Subjects 01 and 02

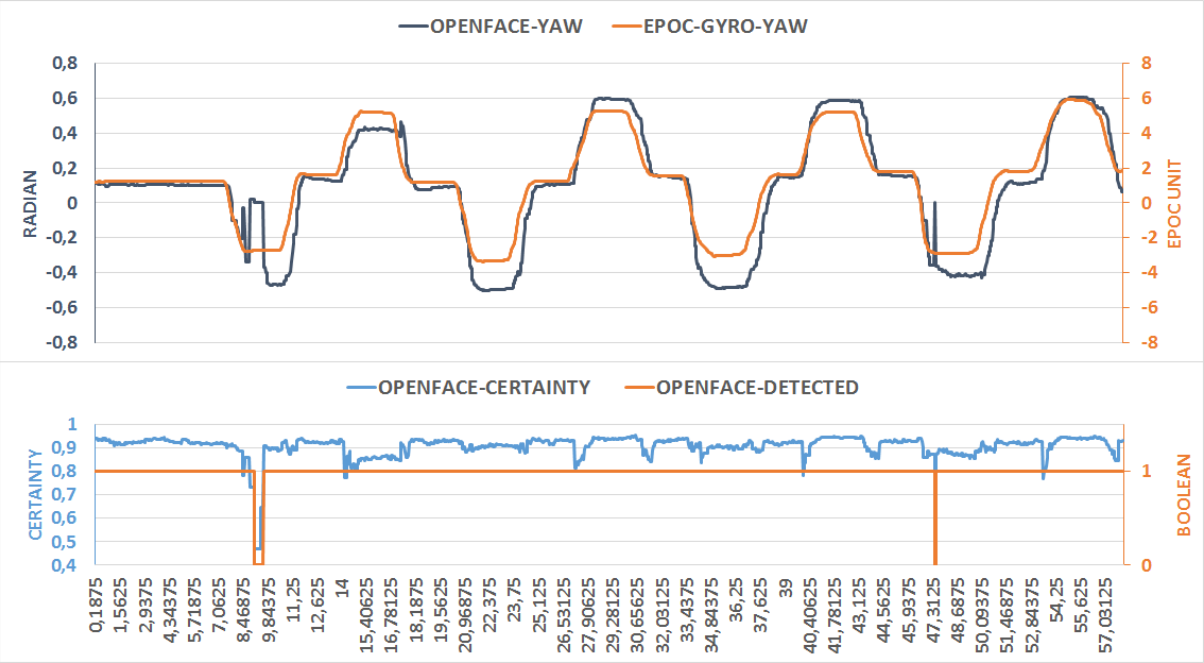


**Figure A.9:** Experiment 2 Data of Subject 01

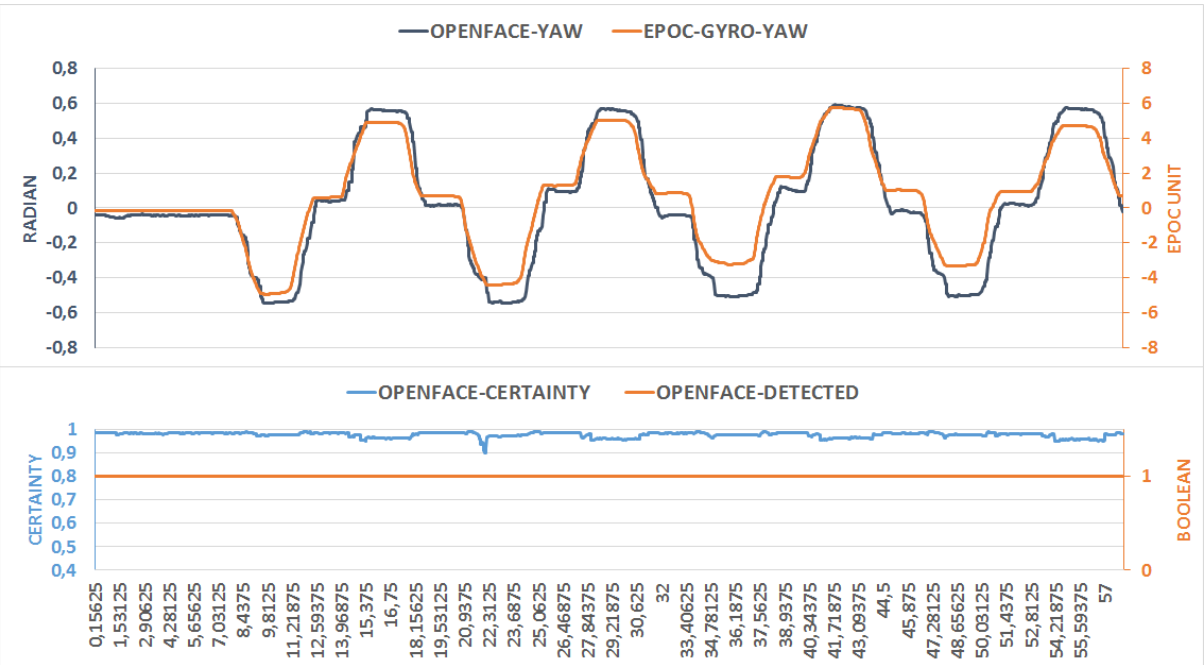


**Figure A.10:** Experiment 2 Data of Subject 02

## A.2.2 Subjects 03 and 04

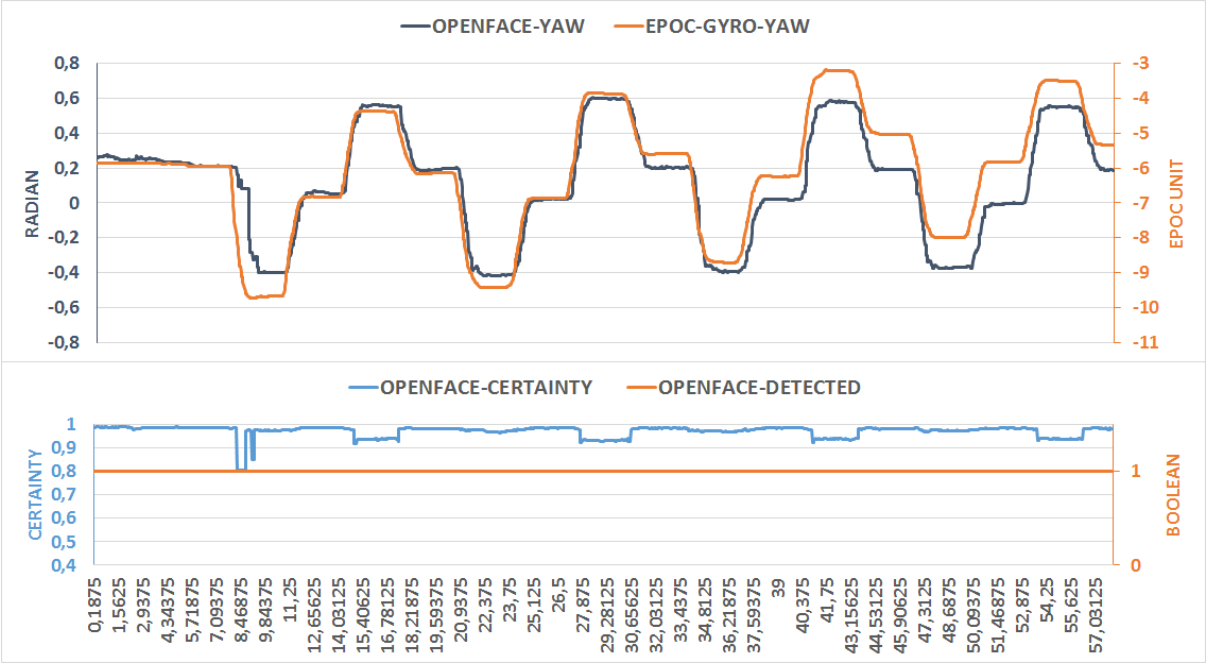


**Figure A.11:** Experiment 2 Data of Subject 03

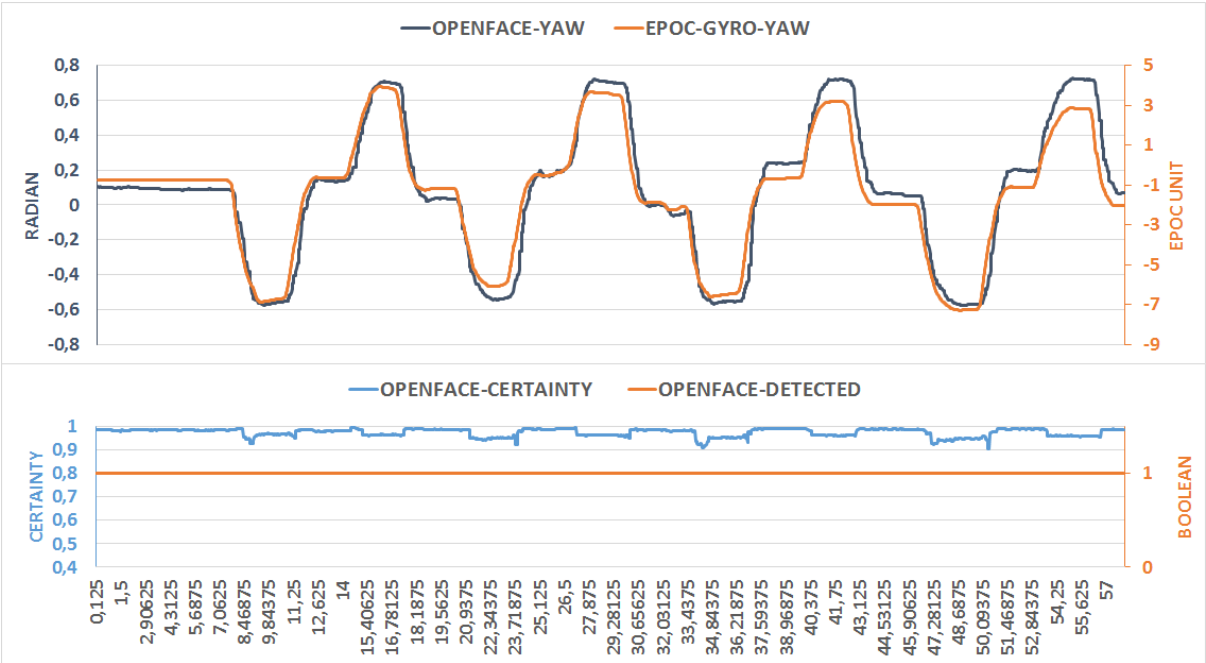


**Figure A.12:** Experiment 2 Data of Subject 04

### A.2.3 Subjects 05 and 06

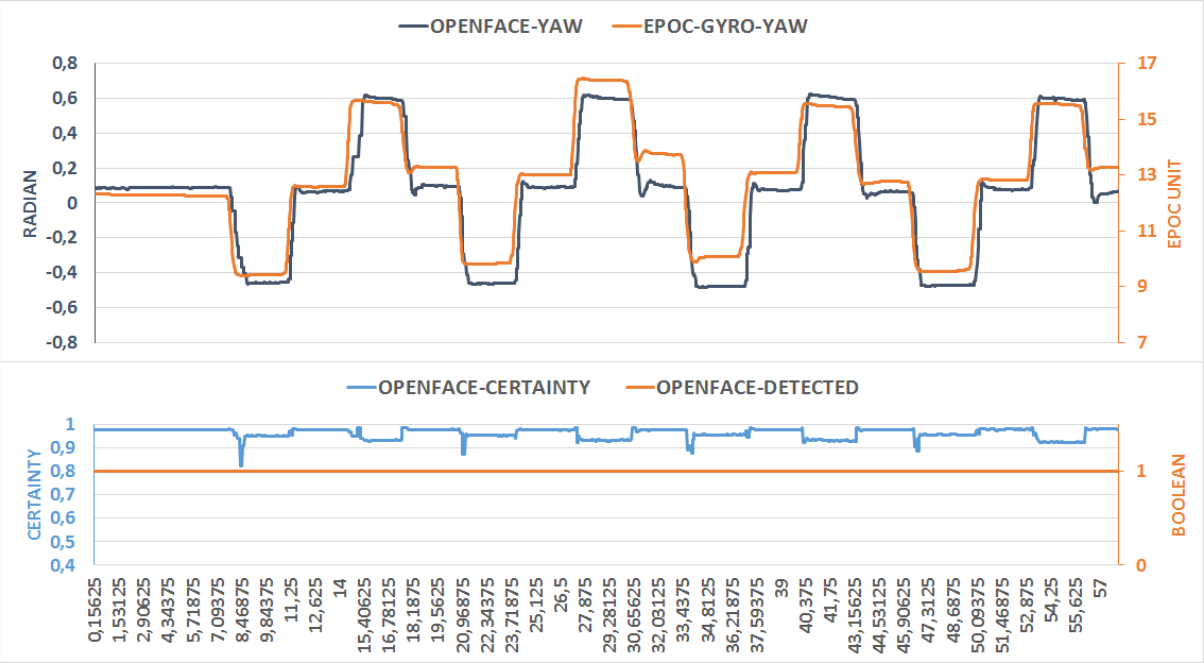


**Figure A.13:** Experiment 2 Data of Subject 05

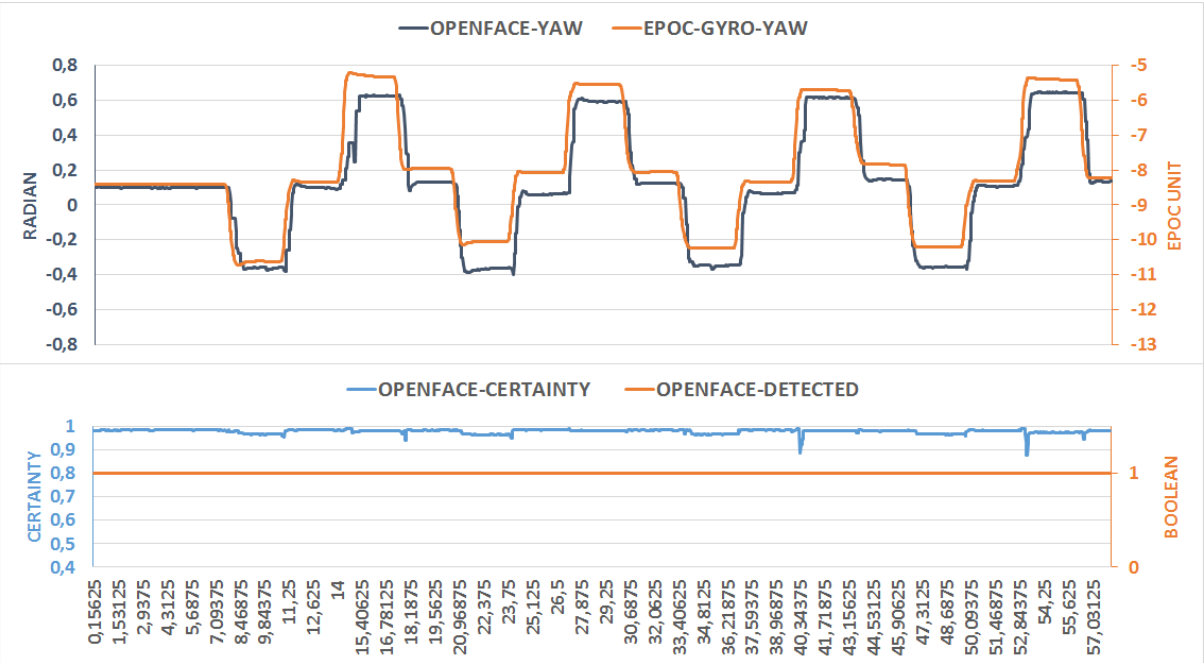


**Figure A.14:** Experiment 2 Data of Subject 06

### A.2.4 Subjects 07 and 08

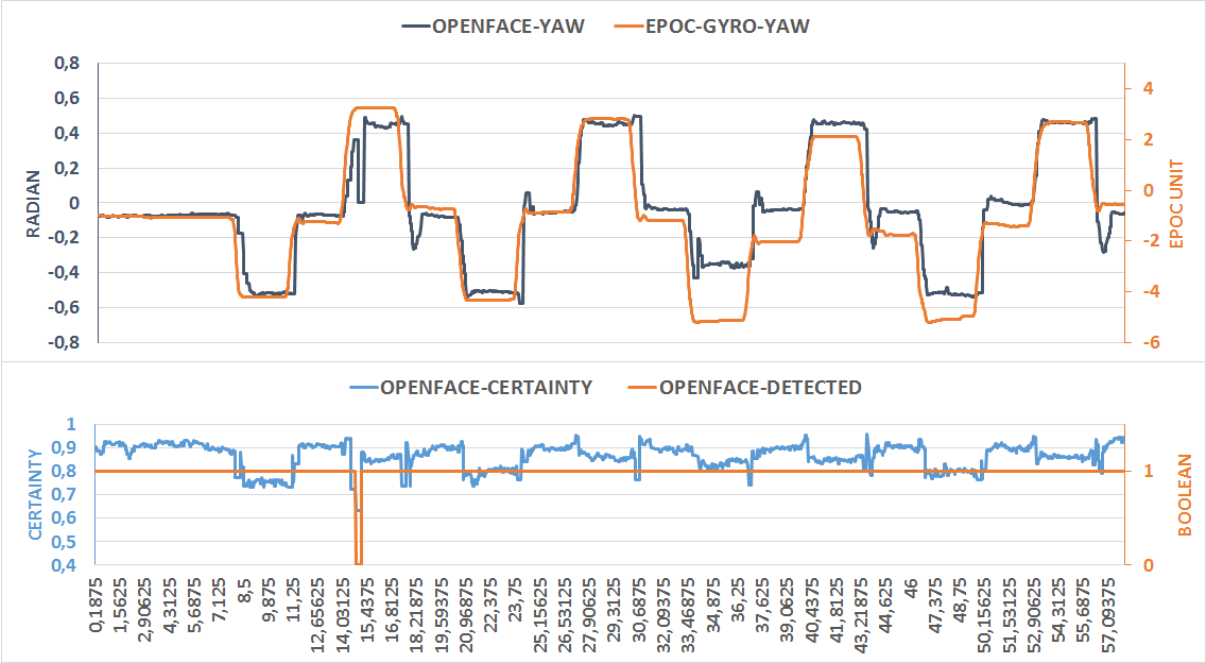


**Figure A.15:** Experiment 2 Data of Subject 07

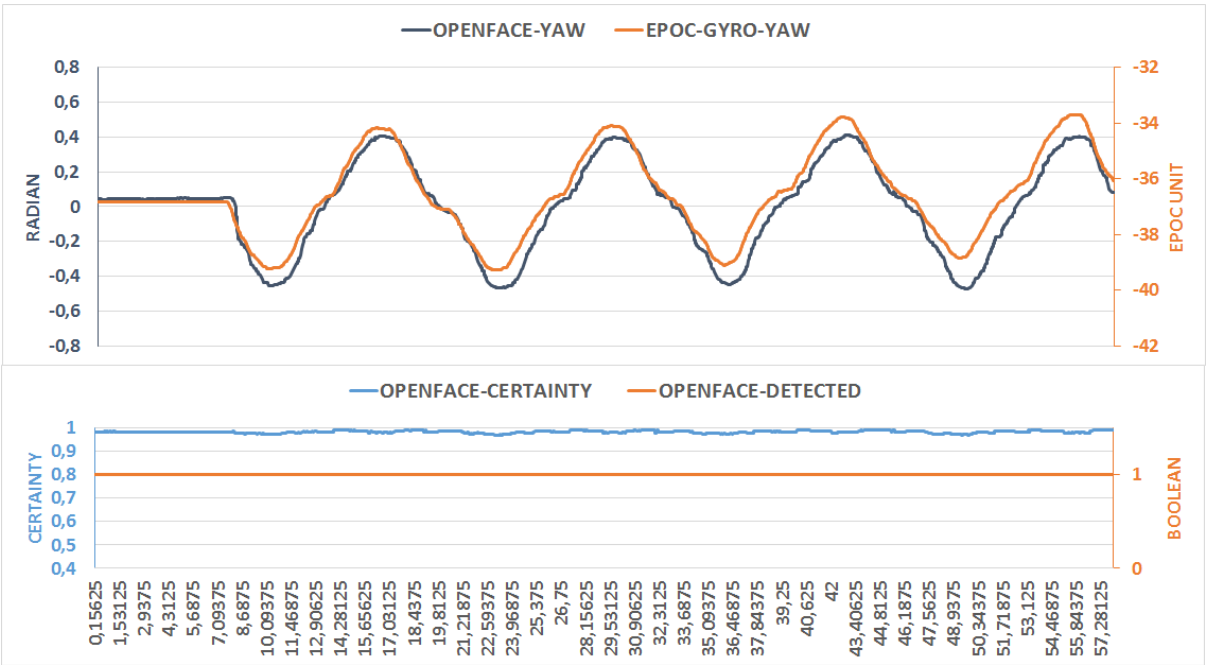


**Figure A.16:** Experiment 2 Data of Subject 08

## A.3 Experiment 3

Visualizations of all data sets that were captured and investigated in *Experiment 3* are presented on the next pages.

For further details about that experiment please refer to section 4.5.

### A.3.1 Subject 01

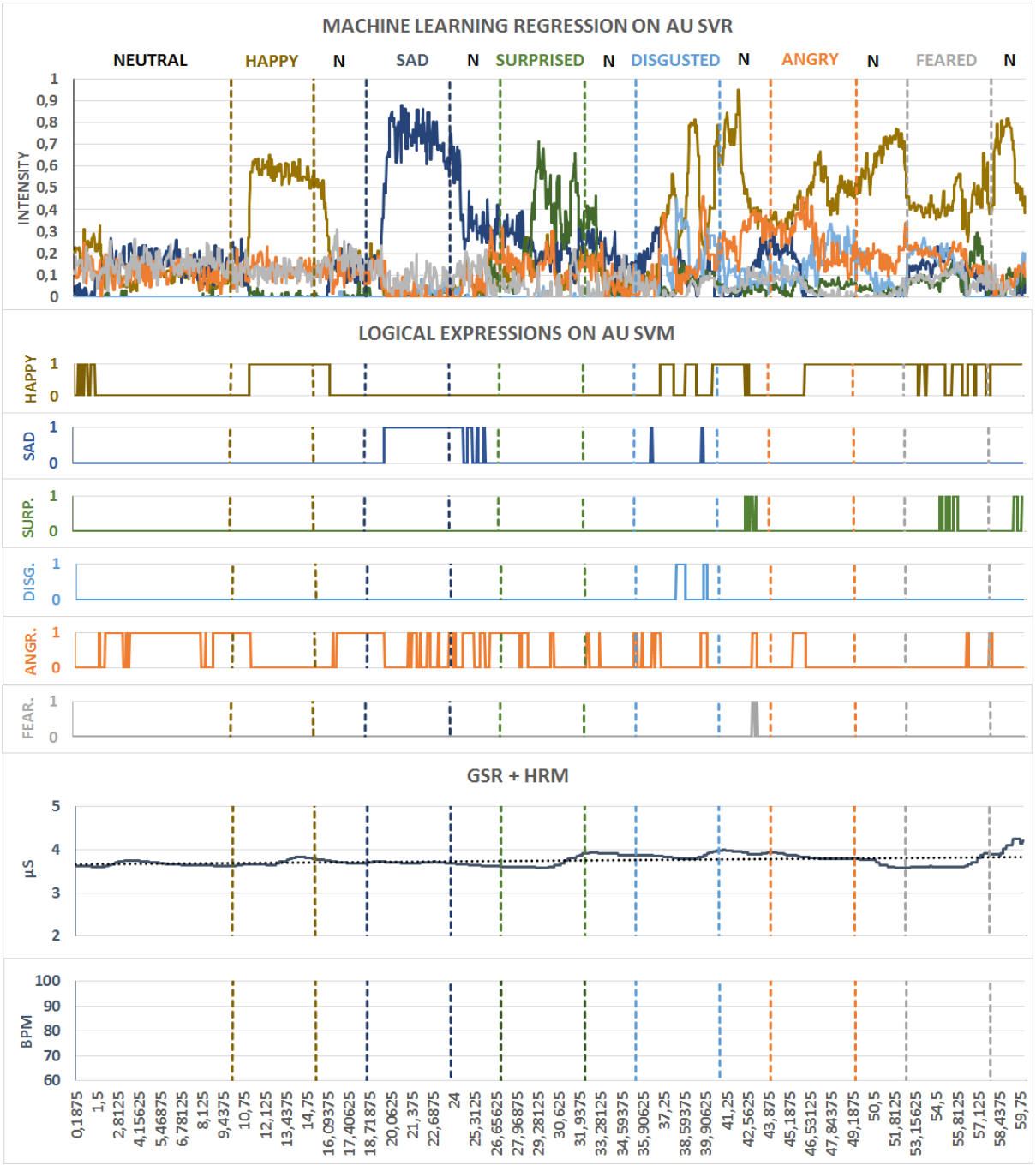


**Figure A.17:** Experiment 3 Data of Subject 01

## A.3.2 Subject 02

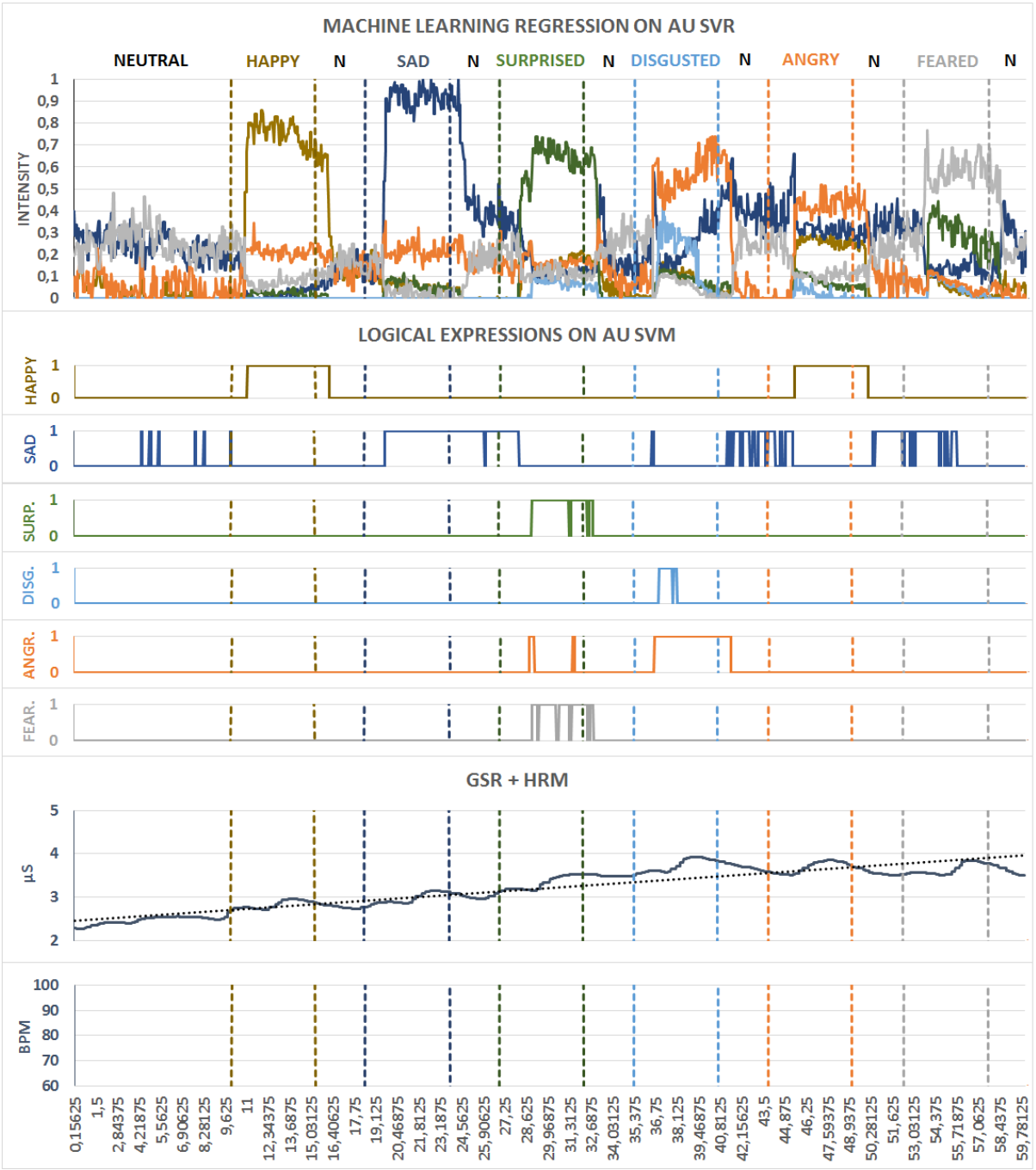


**Figure A.18:** Experiment 3 Data of Subject 02

### A.3.3 Subject 03

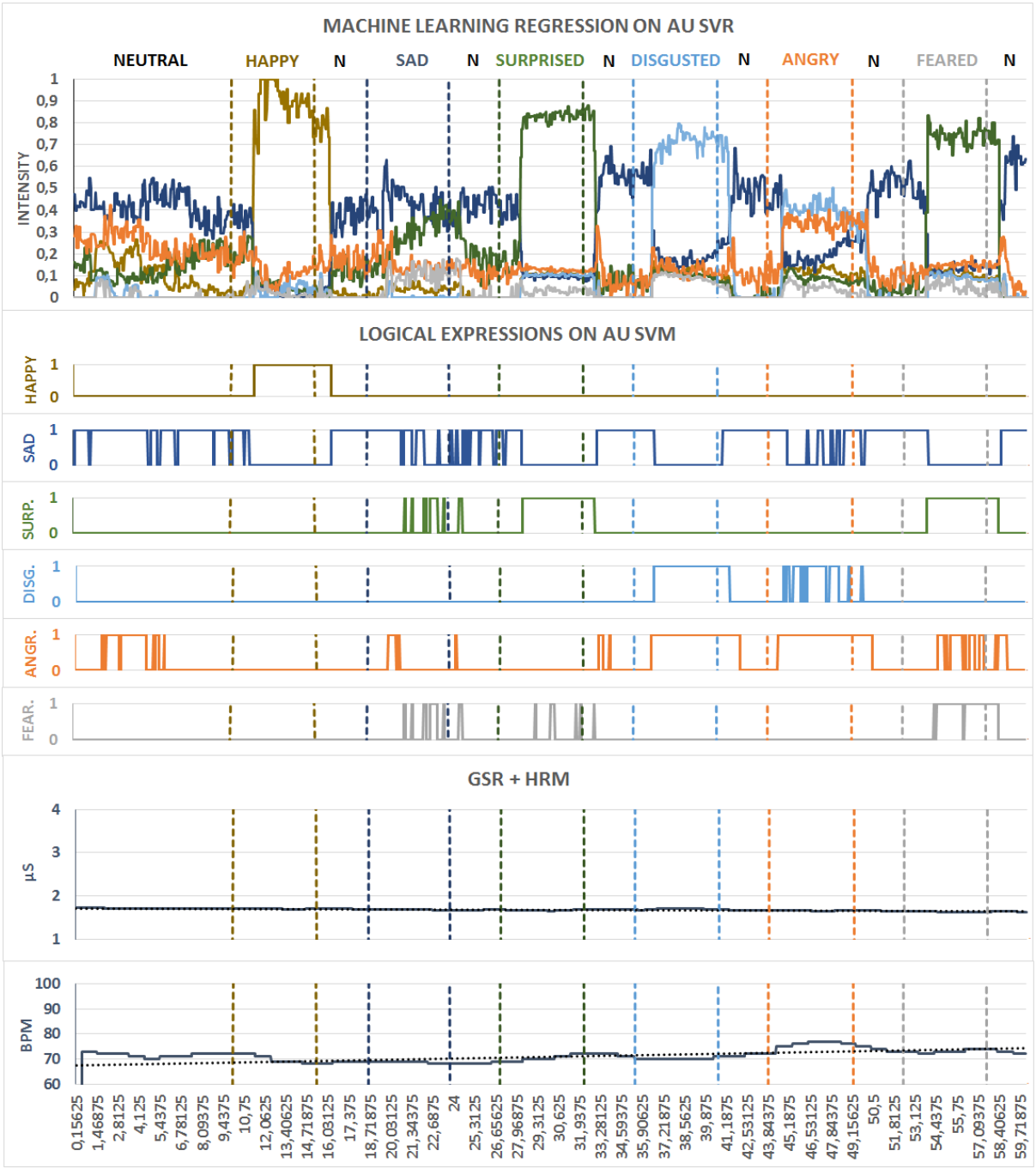


**Figure A.19:** Experiment 3 Data of Subject 03

### A.3.4 Subject 04

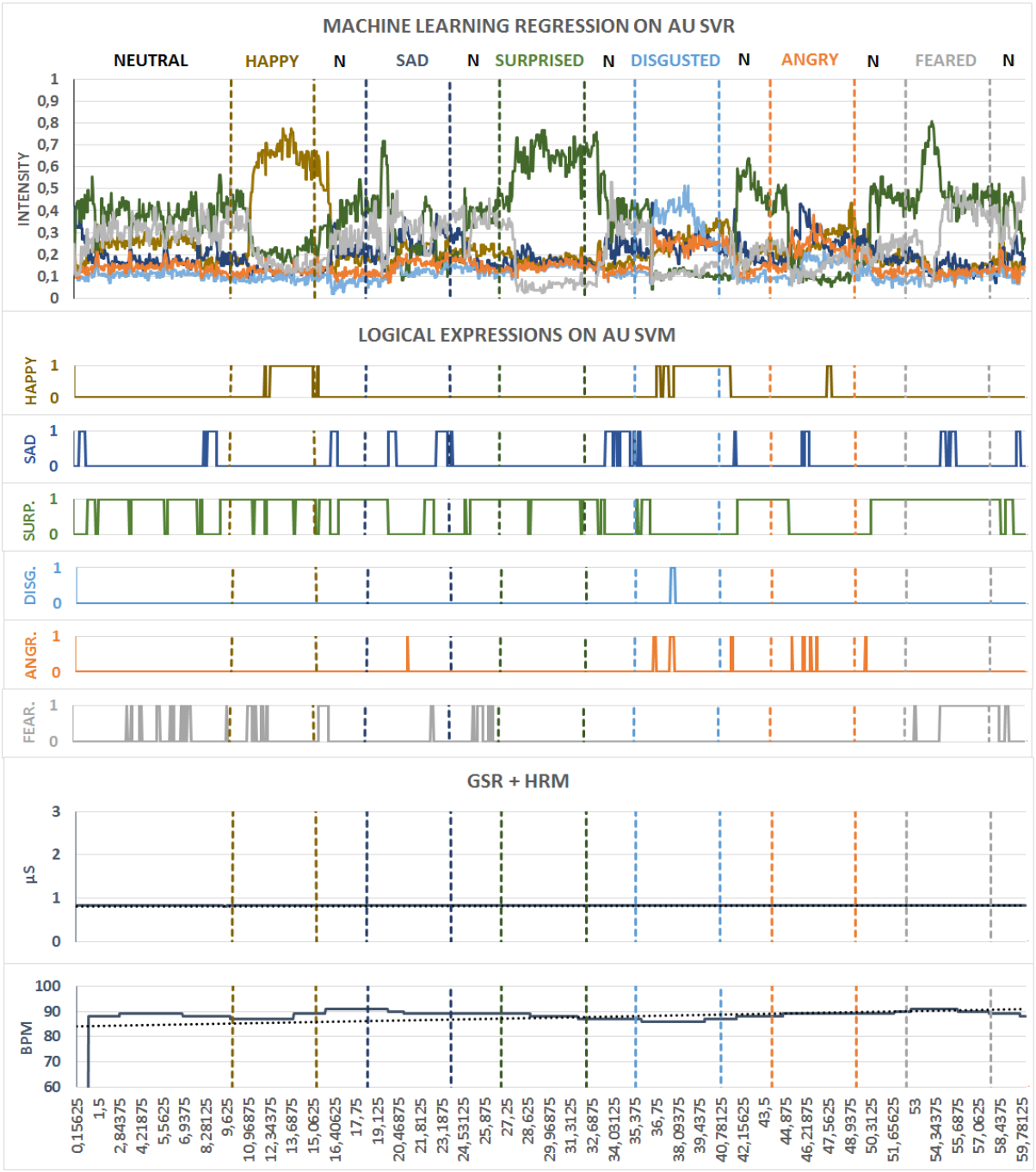


**Figure A.20:** Experiment 3 Data of Subject 04

### A.3.5 Subject 05

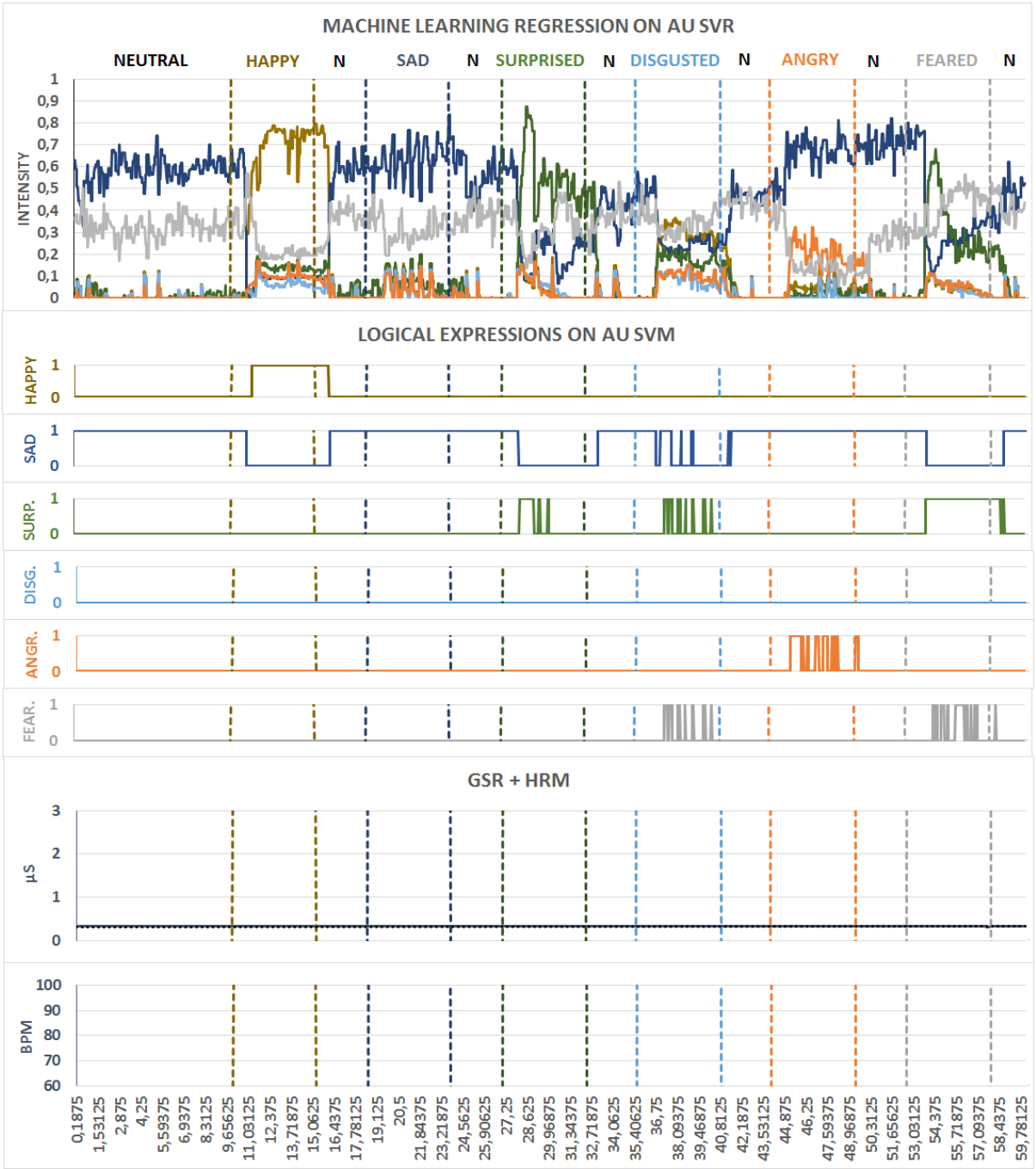


**Figure A.21:** Experiment 3 Data of Subject 05

## A.3.6 Subject 06

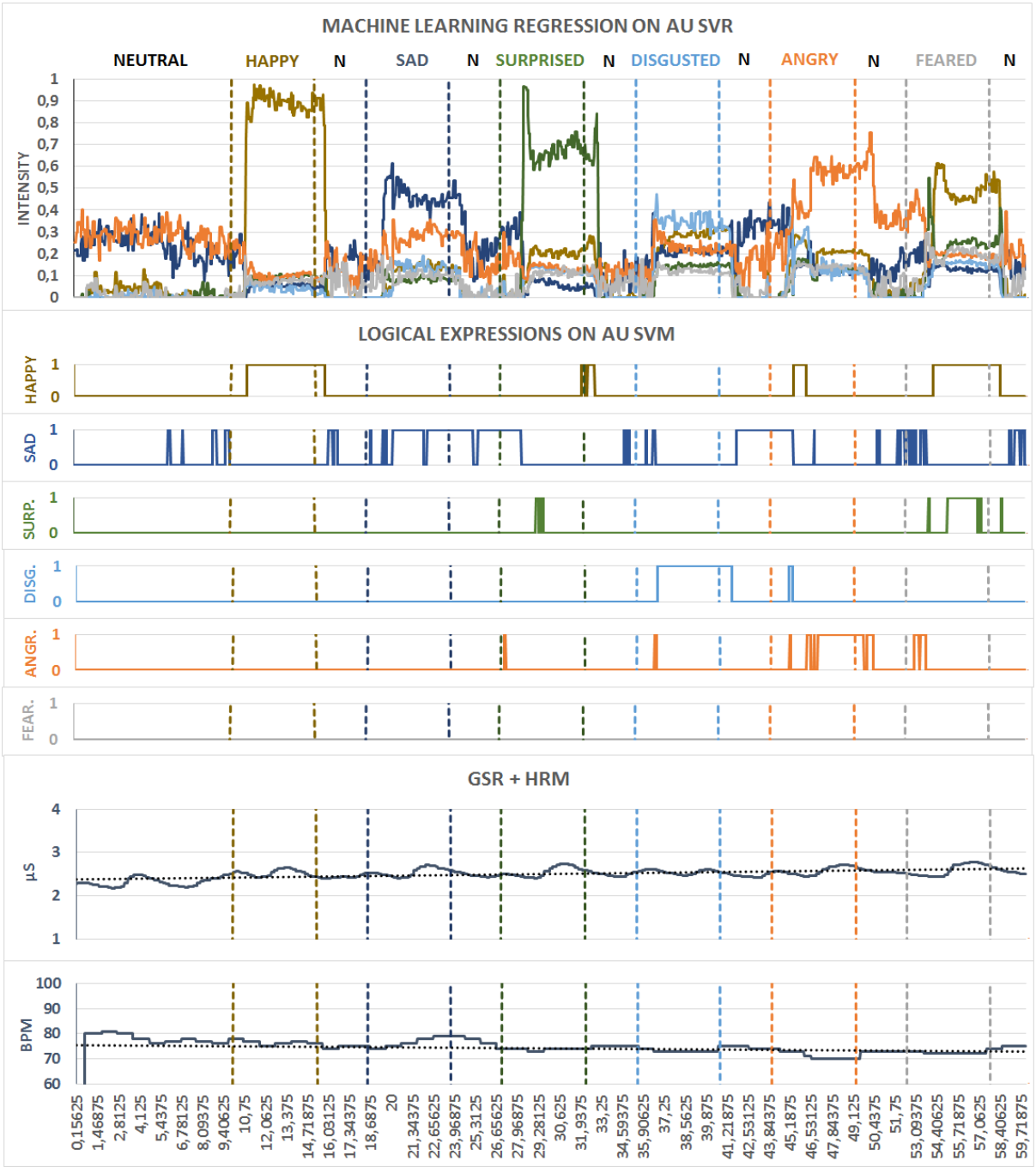


**Figure A.22:** Experiment 3 Data of Subject 06

### A.3.7 Subject 07

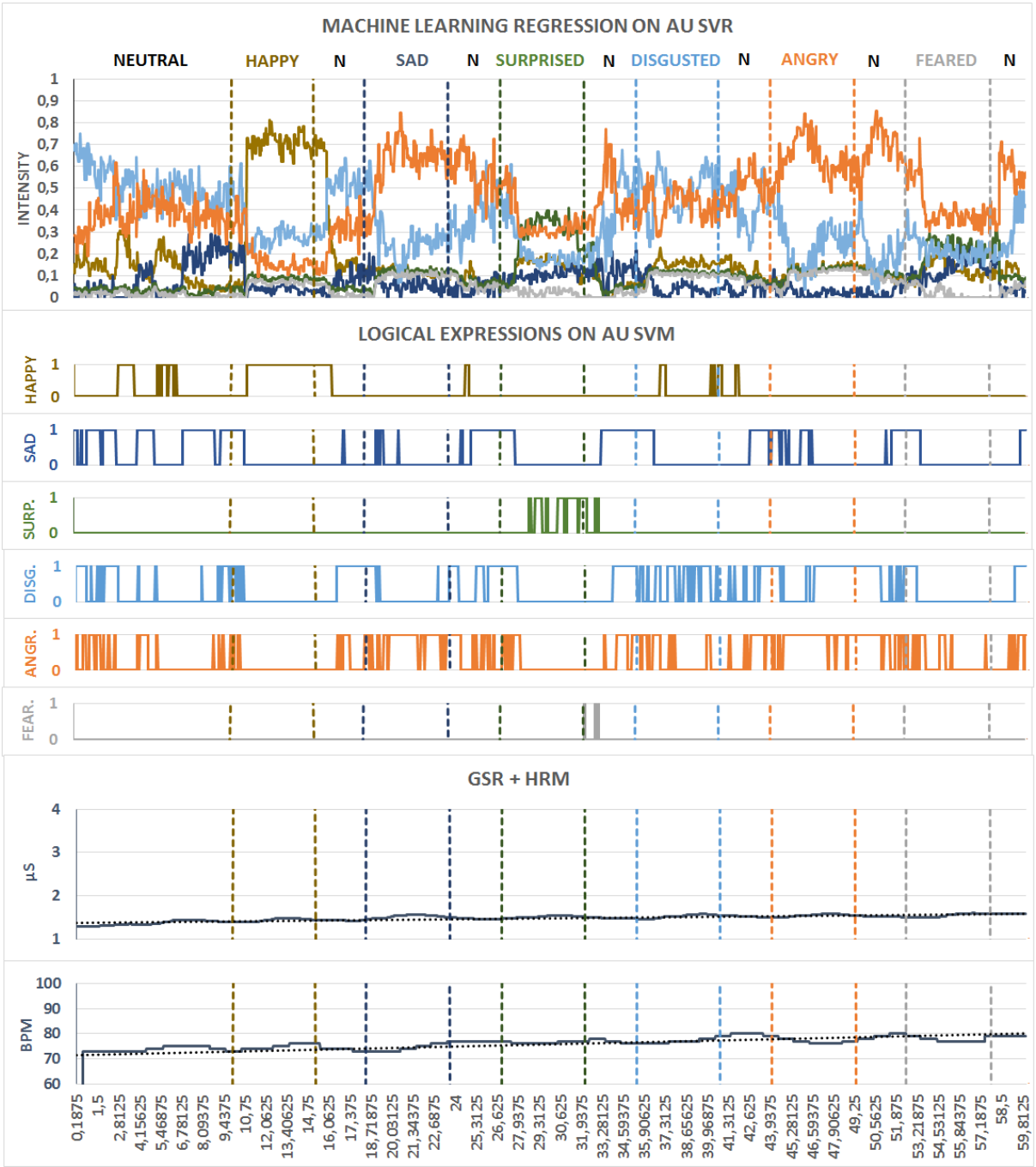


**Figure A.23:** Experiment 3 Data of Subject 07

## A.3.8 Subject 08

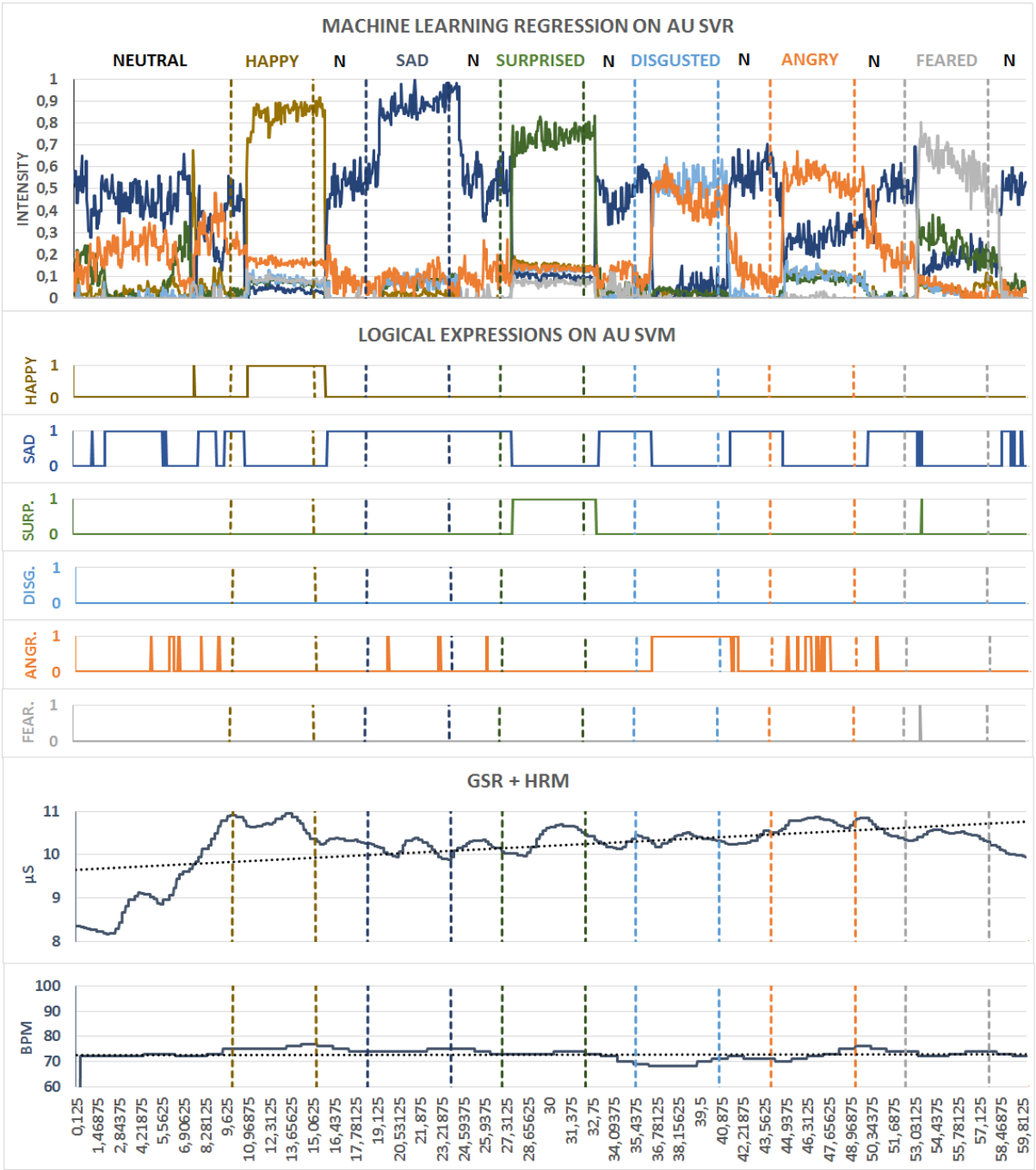


**Figure A.24:** Experiment 3 Data of Subject 08

# Bibliography


[300W13] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. "300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge." In: *IEEE Int'l Conf. on Computer Vision, (ICCV-W), 300 Faces in-the-Wild Challenge (300-W), Sydney, Australia, 2013*. 2013 (cit. on p. 23).

[300W16] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. "300 Faces In-The-Wild Challenge." In: *Image Vision Comput.* 47.C (Mar. 2016), pp. 3–18. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2016.01.002. URL: http://dx.doi.org/10.1016/j.imavis.2016.01.002 (cit. on p. 23).

[AFLW11] M. Köstinger, P. Wohlhart, P. M. Roth, H. Bischof. *Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization*. 2011. URL: https://lrs.icg.tugraz.at/research/aflw/ (cit. on pp. 23, 27).

[Ahonen04] T. Ahonen, A. Hadid, M. Pietikäinen. "Face Recognition with Local Binary Patterns." In: *Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part I*. Ed. by T. Pajdla, J. Matas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 469–481. ISBN: 978-3-540-24670-1. DOI: 10.1007/978-3-540-24670-1_36. URL: http://dx.doi.org/10.1007/978-3-540-24670-1_36 (cit. on p. 23).

[ASM95] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham. "Active Shape Models&Mdash;Their Training and Application." In: *Comput. Vis. Image Underst.* 61.1 (Jan. 1995), pp. 38–59. ISSN: 1077-3142. DOI: 10.1006/cviu.1995.1004. URL: http://dx.doi.org/10.1006/cviu.1995.1004 (cit. on p. 34).

[ATT92] M. Pantic, M. Valstar, R. Rademaker, L. Maat. *Parameterisation of a Stochastic Model for Human Face Identification*. 1992. URL: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html (cit. on p. 27).

[BU-3DFE06] L. Yin, X. Wei, Y. Sun, J. Wang, M. J. Rosato. *A 3D Facial Expression Database For Facial Behavior Research*. 2006. URL: http://www.cs.binghamton.edu/~lijun/Research/3DFE/3DFE_Analysis.html (cit. on p. 27).

[CCNF14] T. Baltrušaitis, P. Robinson, L.-P. Morency. "Continuous Conditional Neural Fields for Structured Regression." In: *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV*. Ed. by D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars. Cham: Springer International Publishing, 2014, pp. 593–608. ISBN: 978-3-319-10593-2. DOI: 10.1007/978-3-319-10593-2_39. URL: http://dx.doi.org/10.1007/978-3-319-10593-2_39 (cit. on p. 32).

[CK+10] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, I. Matthews. *The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression*. 2010. URL: http://www.pitt.edu/~emotion/ck-spread.htm (cit. on p. 27).

[CLandmark15] M. Uřičář, V. Franc, D. Thomas, S. Akihiro, V. Hlaváč. "Real-time Multi-view Facial Landmark Detector Learned by the Structured Output SVM." In: *11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2015*. Vol. 02. 2015, pp. 1–8. DOI: 10.1109/FG.2015.7284810 (cit. on p. 34).

[clmtrackr11] J. M. Saragih, S. Lucey, J. F. Cohn. "Deformable Model Fitting by Regularized Landmark Mean-Shift." In: *Int. J. Comput. Vision* 91.2 (Jan. 2011), pp. 200–215. ISSN: 0920-5691. DOI: 10.1007/s11263-010-0380-4. URL: http://dx.doi.org/10.1007/s11263-010-0380-4 (cit. on p. 33).

[CLMZ12] T. Baltrušaitis, P. Robinson, L.-P. Morency. "3D Constrained Local Model for Rigid and Non-rigid Facial Tracking." In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2610–2617. ISBN: 978-1-4673-1226-4. URL: http://dl.acm.org/citation.cfm?id=2354409.2354947 (cit. on p. 32).

[CLNF13] T. Baltrušaitis, P. Robinson, L.-P. Morency. "Constrained Local Neural Fields for Robust Facial Landmark Detection in the Wild." In: *Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops*. ICCVW '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 354–361. ISBN: 978-1-4799-3022-7. DOI: 10.1109/ICCVW.2013.54. URL: http://dx.doi.org/10.1109/ICCVW.2013.54 (cit. on p. 32).

[DISFA13] S. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, J. F. Cohn. *DISFA: A Spontaneous Facial Action Intensity Database*. 2013. URL: http://www.engr.du.edu/mmahoor/DISFA.htm (cit. on p. 27).

[dlib09] D. E. King. "Dlib-ml: A Machine Learning Toolkit." In: *Journal of Machine Learning Research* 10 (2009), pp. 1755–1758 (cit. on p. 31).

[EEG69] P. Gloor. "Hans Berger on Electroencephalography." In: *American Journal of EEG Technology* 9.1 (1969), pp. 1–8. DOI: 10.1080/00029238.1969.11080728. eprint: http://www.tandfonline.com/doi/pdf/10.1080/00029238.1969.11080728. URL: http://www.tandfonline.com/doi/abs/10.1080/00029238.1969.11080728 (cit. on p. 28).

[Eigenfaces91] M. Turk, A. Pentland. "Eigenfaces for Recognition." In: *J. Cognitive Neuroscience* 3.1 (Jan. 1991), pp. 71–86. ISSN: 0898-929X. DOI: 10.1162/jocn.1991.3.1.71. URL: http://dx.doi.org/10.1162/jocn.1991.3.1.71 (cit. on p. 23).

[FACS78] P. Ekman, W. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. 1978 (cit. on p. 24).

[Fisherfaces97] P. N. Belhumeur, J. a. P. Hespanha, D. J. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 19.7 (July 1997), pp. 711–720. ISSN: 0162-8828. DOI: 10.1109/34.598228. URL: http://dx.doi.org/10.1109/34.598228 (cit. on p. 23).

[Flandmark12] M. Uřičář, V. Franc, V. Hlaváč. "Detector of Facial Landmarks Learned by the Structured Output SVM." In: *VISAPP '12: Proceedings of the 7th International Conference on Computer Vision Theory and Applications*. (Rome, Italy). Ed. by G. Csurka, J. Braz. Vol. 1. Portugal: SciTePress — Science and Technology Publications, 2012, pp. 547–556. ISBN: 978-989-8565-03-7 (cit. on p. 34).

[Helen12] V. Le, J. Brandt, Z. Lin, L. Bourdev, T. S. Huang. *Interactive Facial Feature Localization*. 2012. URL: http://www.ifp.illinois.edu/~vuongle2/helen/ (cit. on pp. 23, 27).

[HOG-SVM05] N. Dalal, B. Triggs. *Histograms of Oriented Gradients for Human Detection*. 2005 (cit. on pp. 21, 22, 31, 42).

[JAFFE98] M. Lyons, S. Akamatsu, Soraku-gun. "Coding Facial Expressions with GaborWavelets." In: *Proceedings, Third IEEE International Conference on Automatic Face and Gesture Recognition*. 1998, pp. 200–205. URL: http://www.kasrl.org/jaffe.html (cit. on p. 27).

[Kanade73] T. Kanade. *Picture Processing System by Computer Complex and Recognition of Human Faces*. 1973 (cit. on p. 21).

[LFW07] G. B. Huang, M. Ramesh, T. Berg, E. Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. 2007. URL: http://vis-www.cs.umass.edu/lfw/ (cit. on p. 27).

[libsvm11] C.-C. Chang, C.-J. Lin. "LIBSVM: A library for support vector machines." In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27 (cit. on p. 31).

[MMI05] M. Pantic, M. Valstar, R. Rademaker, L. Maat. *WEB-BASED DATABASE FOR FACIAL EXPRESSION ANALYSIS*. 2005. URL: http://mmifacedb.eu/ (cit. on p. 27).

[MUG10] N. Aifanti, C. Papachristou, A. Delopoulos. *The MUG facial expression database*. 2010. URL: http://mug.ee.auth.gr/fed/ (cit. on p. 27).

[Multi-PIE08] R. Gross, I. Matthews, J. Cohn, T. Kanade, S. Baker. *Multi-PIE*. 2008. URL: http://www.flintbox.com/public/project/4742/ (cit. on pp. 23, 27).

[OpenCV15] Itseez. *Open Source Computer Vision Library*. 2015. URL: http://opencv.org/ (cit. on p. 30).

[OpenFace16] T. Baltrušaitis, P. Robinson, L.-P. Morency. "OpenFace: an open source facial behavior analysis toolkit." In: *IEEE Winter Conference on Applications of Computer Vision*. 2016. URL: https://github.com/TadasBaltrusaitis/OpenFace (cit. on pp. 32, 37).

[Osuna97] E. Osuna, R. Freund, F. Girosi. *Training Support Vector Machines: an Application to Face Detection*. 1997 (cit. on p. 21).

[Rowley96] H. A. Rowley, S. Baluja, T. Kanade. *Neural Network-Based Face Detection*. 1996 (cit. on p. 21).

[stasm14] S. Milborrow, F. Nicolls. "Active Shape Models with SIFT Descriptors and MARS." In: *VISAPP* (2014) (cit. on p. 34).

[SVM95] C. Cortes, V. Vapnik. "Support-vector networks." In: *Machine Learning* 20.3 (1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: http://dx.doi.org/10.1007/BF00994018 (cit. on p. 20).

[ViolaJones01] P. Viola, M. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*. 2001 (cit. on pp. 21, 22, 30, 33, 34, 42).

[YaleA97] *Yale Face Database A*. 1997. URL: http://vision.ucsd.edu/content/yale-face-database (cit. on p. 27).

[YaleB+05] K.-C. Lee, J. Ho, D. Kriegman. *Acquiring Linear Subspaces for Face Recognition under Variable Lighting*. 2005. URL: http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html (cit. on p. 27).

All links were last followed on January 18, 2017.

**Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature