SEVENTH FRAMEWORK PROGRAMME

AsTeRICS Deliverable D4.6a

Final Prototype of Signal Processing Modules – Algorithms

UPMC

AsTeRICS

European Commission
Information Society and Media

Document Information

Issue Date	**30 June 2012**
Deliverable Number	**D4.6**
WP Number	**WP4 Software**
Status	**Final**
Dissemination Level	**CO** **PU Public** **PP Restricted to other programme participants (including the Commission Services)** **RE Restricted to a group specified by the consortium (including the Commission Services)** **CO Confidential, only for members of the consortium (including the Commission Services)**

AsTeRICS – Assistive Technology Rapid Integration & Construction Set
Grant Agreement No.247730
ICT-2009.7.2 - Accessible and Assistive ICT
Small or medium-scale focused research project

Disclaimer

Version History

Version	Date	Changed	Author(s)
0	05/06/2012	First draft	Chris Veigl
0.1	12/06/2012	Initial edit of Starlab sections	Aureli Soria-Frisch
1.0	12/06/2012	Updated TOBI integration section	Aureli Soria-Frisch
1.1	18/06/2012	TOBI internal review and trigger section	Javier Acedo
1.2	19/06/2012	Updated and BNCI p300	Aureli Soria-Frisch
1.3	20/06/2012	Update OpenVIBE Starlab and SSVEP functionalities	David Ibáñez, Anton Albajes-Eizagirre
1.4	25/06/2012	Updated SVM, including FHTW contributions	A. Carbone
1.5	28/06/2012	Included peer review comments and updates	A. Carbone

Table of Content

Glossary

ACS	AsTeRICS Configuration Suite
ADC	Analogue to Digital Converter
ARE	AsTeRICS Runtime Environment
ASM	Active Shape Model
CIM	Communication Interface Module
CSP	Common Spatial Patterns
DAC	Digital to Analogue
EEG	Electroencephalogram
EMG	Electromyogram
EOG	Electrooccculogram
FPR	False Positive Rate
GPIO	General Purpose Input and Output
HID	Human Interface Device
IMU	Inertial Measurement Unit
KNN	K-nearest neighbour
KNX	Konnex Home Automation Standard
LDA	Linear discriminant analysis
OSKA	On Screen Keyboard Application
OVR	One versus the rest
PR	Precision-Recall
ROC	Receiver Operating Curve
SVM[1]	Smart Vision Module / Support Vector Machine
SVM	Smart Vision Module
TPR	True Positive Rate

[1] Unfortunately there are two different meanings associated to the SVM shortcut. We made our best to disambiguate which is the meaning in each section of the text.

1 Introduction

Deliverable 4.6 describes accomplished works within AsTeRICS project's WP4, which includes software implementation activities. Particularly, the deliverable describes the development of the final AsTeRICS system prototype (Prototype-2) within the following tasks according to the Description of Work (see [1], page 58):

- T4.8 - Signal Processing Framework development and refinement for final Prototype
- T4.9 - Video Signal Processing Components development and refinement for final Prototype
- T4.10 - Development and refinement of Software toolkit for BCI based on machine learning for final Prototype

Due to the strongly heterogeneous contents of D4.6 and the recommendation of the reviewers of deliverable D4.3 (signal processing modules for Prototype-1), D4.6 is split into two parts, which considerably improves the consistency and readability of the deliverable:

- D4.6a – *Algorithms*, (this document) which focuses on the scientific work in algorithms for BNCI and Computer Vision (SVM) and covers the tasks T4.9 and T4.10
- D4.6b – *Plugins*, which describes the software engineering work for ARE-plugins and AsTeRICS models and covers task T4.8

The activity T4.8 includes the adaptation of existing plugins and signal processing functionalities for the AsTeRICS Runtime Environment (ARE), and the development of new plugins according to the results of the user evaluation of Prototype-1. Furthermore, several new plugins had to be developed to provide the intended features of the final system prototype. As in Prototype-1, the basic purpose of the developed plugins is to be included in models for the realization of more complex tasks, which can be configured via the AsTeRICS Configuration Suite (ACS). Section 3 of this deliverable shows several examples of AsTeRICS models using some of the previously described plugins..

Within task T4.9, the Computer Vision algorithms and strategies for the Smart Vision Module (SVM) were refined and evolved. These efforts include the improvement of the webcamera-based ("remote") head tracking algorithms and the development of a new head-mounted camera system which is capable of head-tracking, iris-tracking and gaze estimation. A dedicated hardware for the head-mounted system has been developed which consists of a microcontroller-board, inertial measurement unit (IMU) and 3d-printed mounting parts. The available SVM-features for the final system include head pose estimation, recognition of facial gestures, eye blink detection and integration of inertial sensing data.

Activities in task T4.10 cover the works related to Brain-Neural-Computer Interfaces, respectively the finalisation of the BNCI toolkit and its application. These works have been focused in the BCI paradigms of motor imagery and p300, and SSVEP where special hardware for visual stimulation has been developed. Additionally to integration with BCI2000 [4] and BioSig [3], a complete OpenVibe [5] support for the Enobio system and a bridge from OpenVibe to AsTeRICS has been developed, and interoperability of BNCI functionalities with the FP7-cofunded "TOBI" project [2] have been evaluated.

2 BNCI Evaluation Suite

In the following section we report on the different modules implemented for the Evaluation Suite and different BNCI related works undertaken within AsTeRICS WP4, T4.10. This task aims at the development of several functionalities around the Enobio sensor for the realization of BCI functionalities, which should flow into the AsTeRICS final prototype PT2. The works have taken the initial reviewers recommendation for interfacing existent toolkits and frameworks. In this context we have furthered the integration works with BioSig, BCI2000 and several Matlab toolboxes realized in T4.5 as described in D4.3a [59] by integrating Enobio in OpenVibe. Moreover OpenVibe has been integrated in the overall AsTeRICS platform as described in the following sections. One further focus of attention has been the integration with the implementations resulting from the TOBI workshop, which are discussed as well in the following sections.

In the last few years research on Brain Computer Interface (BCI) has continually increased and this trend seems set to continue [9]. BCI systems attain to translate brain signals, which can include several physiological modalities, into control commands of a particular device, e.g. wheelchair, computer, mobile phone, robot arm [10]. A generic block diagram of such systems can be observed in Figure 1. Taking such a structure as a reference, if a system makes use of a classification procedure as feature translator, the resulting system can be analysed from a pattern recognition perspective. Pattern recognition systems for BCI have been extensively reviewed in [11]. This is the approach we follow in the BNCI Evaluation Suite.

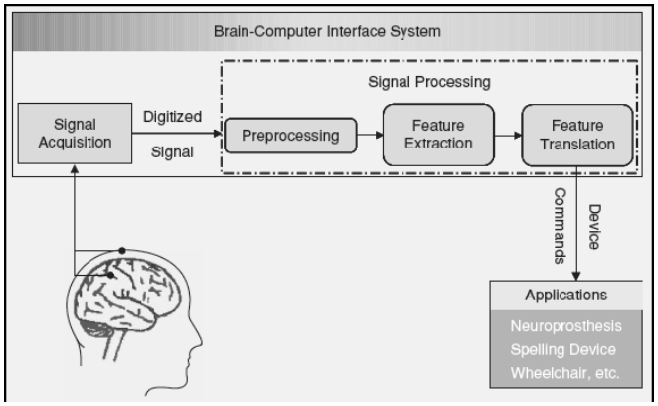


Figure 1. Block diagram of a generic BCI system based on a pattern recognition approach. Reproduced from [10] with permission.

Different paradigms for the realization of BCI exist. Among them the more common used ones are so-called motor imagery, P300, and SSEVP [12]. While the work described in D4.3 [59] was focused on the paradigms motor imagery, and preliminarily on p300 paradigms, we have significantly extended in this iteration the works on p300, and focused on the SSVEP one. However this last work is fully developed within WP6. Therefore we comment here just on the associated methodologies that will be available in the BNCI Evaluation Suite.

One important milestone in the utilization of Enobio for BNCI was the implementation of a new cap and electrode set. This important step was designed, and developed by Starlab outside AsTeRICS work. However it is mentioned here for the sake of completeness.

As it can be observed in the following illustration, the new electrodes and cap allow for the development of the complete set of BNCI paradigms because implementing the 10/20 electrode positioning. An upgraded kit for the Enobio devices has been realized within AsTeRICS works. Such upgraded kits have been provided to 3 AsTeRICS partners.

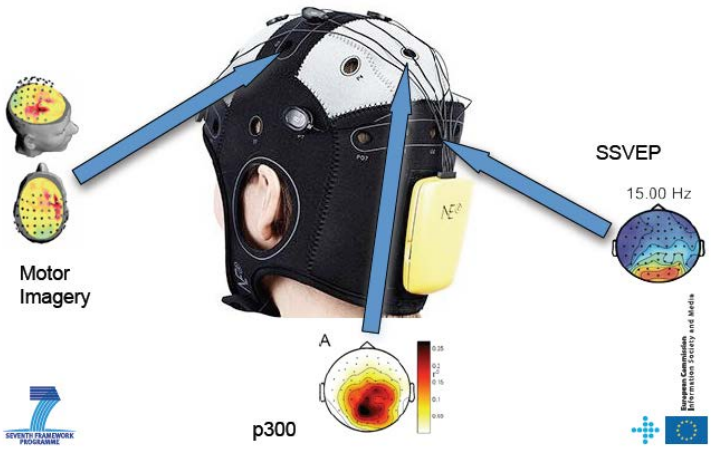


Figure 2. Update of the Enobio cap and electrodes allows the implementation of BNCI applications.

2.1 BCI p300 works

2.1.1 Enobio hardware update for accepting an external trigger

Some of the BCI applications that will run in the second AsTeRICS prototype such as the ones based on the P300 paradigm need a precise synchronization of the EEG signal and an external trigger. The current 4-channel Enobio design does not allow connecting any external signal apart from the 4 electrodes. So some hardware modifications are needed in order to upgrade the device so an external signal could be sent along with the EEG streaming in a safe way.

The following requirements were taken into account for the Enobio hardware update. The resulting device shall behave like the current 4-channel Enobio plus adding the voltage status (low or high) of the external signal to the EEG streaming. The delay in the overall solution shall be in 10 ms order and may be in the 1 ms order. The connection between the external signal and the Enobio box should be made as clean as possible, that is by using some standard connector. The external signal shall be isolated from the circuit for safety reasons.

To include an external signal to the EEG streaming it is necessary to wire it to an input of the microcontroller which is in charge of driving both the wireless link and the serial connection to CPLD that interfaces the EEG electrodes. A direct connection of the external signal to the microcontroller is not allowed since it is needed to isolate both circuitries for safety reasons. The optocoupler has been the default signal isolation device for the past forty years. However, there have recently appeared digital isolators that offer advantages in performance, size, cost, power efficiency and integration. The digital isolators provide a bandwidth above one Megabyte per second so they are not impeding on fulfilling the timing requirements.

The available space on the 4-channel Enobio box is limited to the area compressed on the right-top corner between the charging connector and the battery. There is a rectangle of 16 x 13 millimetres where the hardware solution shall fit. This space is enough to place a board where to solder the isolator chip, the wires from the external connector and the wires to the microcontroller.
The limited space reduces the possibilities to choose the connector. Some examples as a stereo mini jack holder or a mini DIM are from 7 to 14 mm long, so the inclusion of that type of connectors would dramatically reduce the available space for the rest of the circuit. The use of standard pin header connectors is a solution that fits in the available space although the strength of the connection will not be as much as with the other solutions.

The digital isolator ADuM3100[2] from Analog Devices is proposed for isolation of the external signal from the 4- channel Enobio circuitry.

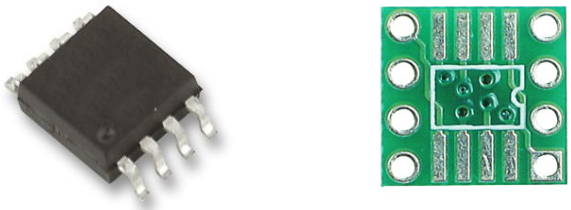


Figure 3. ADuM3100 digital isolator (left) and Pin SOIC to DIP8 board (right).

The 8 Pin SOIC toDIP8 Adapter board from futurlec.com[3] for soldering the ADuM3100 and connect to the external connector and the Enobio board is proposed.

The proposed components fit in the following area within the 4-channel Enobio box:

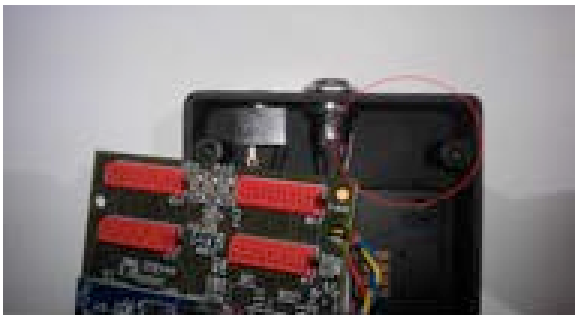


Figure 4. Available space at 4-channel Enobio box.

Once the 8 Pin SOIC to DIP8 Adapter board is placed and fixed to the box the orientation of the header connector will be like the one in the figure above, the pin on the right side being the PIN1 on the schematic so the Vdd signal and the one on the left the PIN4 so the GND signal. The pin in the middle will be the external signal.

[2] http://www.analog.com/en/interface/digital-isolators/adum3100/products/product.html

[3] http://www.futurlec.com/SMD_Adapters.shtml

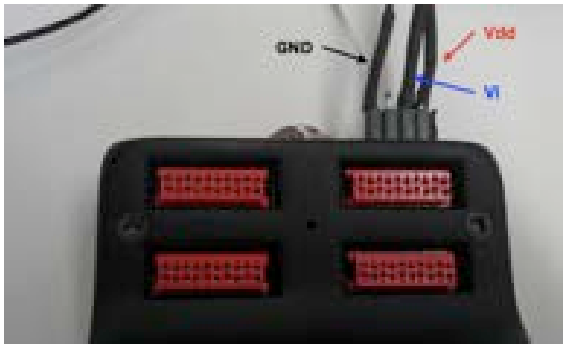


Figure 5. External signal head connector.

Apart from the hardware an update of the firmware that runs on the Enobio microcontroller is needed as well. This update consists of programming the corresponding GPIO pin where the external signal is connected to as an input and set the status of that input (high or low) into the EEG streaming.

2.1.2 Support Vector Machine ensemble for p300 detection

One further task in AsTeRICS was devoted to the development of p300 state-of-the-art for the classification of p300 data. We have focused the works on the spell methodology as published by Rakotomamonjy [37]. The goal of the works is to implement and test the presented methodology. Moreover we are trying to test whether the published schema is susceptible of being improved for the same application of spelling. The final goal is to analyse the feasibility of applying the p300 schema to image search.

We present in this Section the performance evaluation of the data as acquired in the PT1 works and described in [59]. Although the functionalities had been preliminarily integrated, it was necessary to do a refinement of the Evaluation Suite functions in order to really allow such a performance evaluation. The problem with the code as provided in PT1 was that it made use of several internal recorded parameters. Moreover it did not allow analysing the performance for a variable number of epoch averaging. So we have redone the implementation and provide the result of this important work in the Evaluation Suite for PT2.

Once we got a code we could really work with we have conducted a detailed performance evaluation. We have evaluated the performance with two different data sets. First we have used the same data set as in [41]. This was attained in order to ensure we were reproducing the same methodology. Second we have attained the performance evaluation of the methodology on the data set acquired for the AsTeRICS project, which was described in the former deliverable [69]. This data set includes the data of 2 different protocols: the traditional p300 speller and the data generated with a protocol for image browsing. The obtained result analysis will be presented in the following sections.

2.1.2.1 Reproduced p300 methodology

The reproduced methodology presents the stages detailed in the following sections. The Evaluation Suite now integrates functionalities for implementing a system based on these stages. The reader is referred to [41] for a complete description of the framework.

2.1.2.1.1 Data Processing

The data processing for training, validation and classification data is done as follows.

- Cutting: According to the literature the evoked potential appears about 300ms after the stimulus. The data samples are extracted between 0 to 667ms after the beginning of stimuli onset. It is supposed that this window is large enough to capture all required time features for an efficient classification.
- Filter: Each extracted signal is filtered with an 8-order band-pass Chebyshev Type I filter with cut-off frequencies of 0.1 and 10 Hz.
- Resample: Data is decimated according to the high cut-off frequency. Thus the decimation factor is the sampling frequency divided by the filter's high cut-off frequency. At this point the signal from a single channel is composed of 14 samples.

After this processing stage the post-stimulus signals have been transformed into a vector form from the concatenation of all the 64 channels. Therefore the training set is composed of 15300 post-stimulus vectors of dimension 14*64 = 896 samples. Each vector is labelled as post-stimulus vector {1} or non-post-stimulus vector {-1}.

2.1.2.1.2 Split Training Data into Partitions

It is assumed that the Training Data available for each subject is large enough to cluster it in different subsets separating it into training and validation signals for the SVM. The idea is to separate the training signal into homogenous groups. As time chronology of the recorded signals has been lost the simplest approach is to consider that signals related to the spelling of a single character present similar noise features for training the classifier.

The idea is to train and validate 17 SVMs with the available training data. Thus training data has been split into 17 independent partitions. As formerly mentioned we have 85 trials, each one corresponding to a single character spelling. The training data-set has been formed by concatenating 5 trials per partition. The 17 partitions of 5 spelled characters are as follows:

P1 = [T1 T2 T3 T4 T5]
P2 = [T6 T7 T8 T9 T10]
P3 = [T11 T12 T13 T14 T15]
.
.
P17 = [T81 T82 T83 T84 T85]

Because of computational reasons 2 subsets of the training data have been created for the model selection procedure. These 2 subsets group respectively 8 and 9 partition data: SS1 = [P1 P2 P3 P4 P5 P6 P7 P8], and SS2 = [P9 P10 P11 P12 P13 P14 P15 P16 P17]. Therefore SS1 is used to train and validate 8 SVMs and SS2 is used to train and validate 9 SVMs.

2.1.2.1.3 SVM Modelling

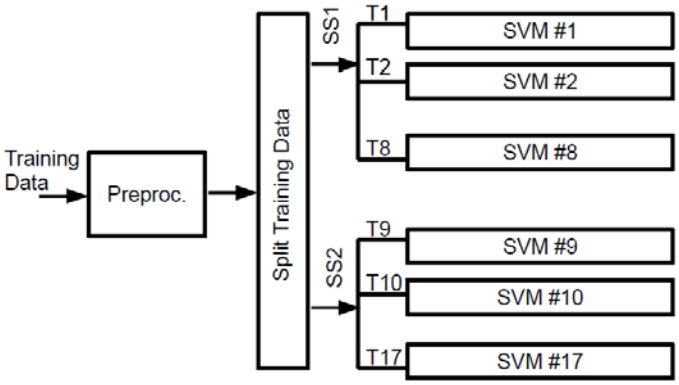


Figure 6. SVM Training Stage Block Diagram.

In this approach ensemble classifiers have been chosen. A 17 classifier system has been designed for each subject, where each classifier is a linear support vector machine trained on one of the 17 partitions. Thus each single classifier of the system is a linear support vector machine trained on one of the 17 partitions. Thus the global system for SVM training is as depicted in Figure 6.

2.1.2.1.4 System parameter selection

In this chapter it is going to be explained how to train and validate the ensemble SVM to obtain its best performance. It is important to mention that SVM performance is evaluated through its corresponding validation signal. The performance of each SVM is evaluated individually and so is it's binary classification performance. The performance of a single SVM is evaluated according to the classification score $C=tp/(tp+fp+fn)$, where:

- tp= True Positive.
- fp= False Positive.
- fn= False Negative.

Model selection procedure involves the choice of SVM hyper-parameter C and the optimal number of channels to use. SVM model parameters are chosen by running a channel selection procedure for different values of C. The pair chosen (channels, C value) is the one that maximizes the classification score. For the selection of C a grid search in [0.01 0.05 0.1 0.5 1] is realized.

For channel selection the following procedure is undertaken. First a single linear SVM is trained with all the features provided by all channels. The Performance of the classifier is evaluated according to the classification score. Then each single channel is temporarily removed and parameter is re-evaluated. The channel whose removal has maximized the score is definitely eliminated. This recursive elimination continues until the number of desired channels is achieved.

2.1.2.2 Performance evaluation BCI competition III data set II

This part of the work has been done with the data set II from the BCI Competition III[4]. The data set has been recorded from 2 different subjects and five different spelling sessions. All EEG signals have been recorded over 64 electrodes at a sampling rate of 240Hz. Signals have been band-pass-filtered from 0.1 to 60Hz.Time chronology has been lost as the data-set provided has been scrambled into trials for a single spelling character. The Training Set provided for each subject has the following characteristics:

- The Training Set is made out of 85 spelling trials each one for a single known character.
- Each trial is made out of 12 character spelling epochs.
- The sequence of intensification is repeated 15 times for each trial.
- Each run has been labelled as post-stimulus or non-post-stimulus run.

Results have been tested over 100 spelling characters for 15 intensification sequences for two subjects. Classification performance has been evaluated in % of correctly recognized characters for the 2 subjects.

The following tests have been carried out in order to individually test the performance of each functionality included in the Evaluation Suite. We give as well the obtained classification score for each subject:

1. Testing the performance of the classification function: Pre-processed data and the Support Vector Machines of the BCI III P300 Spelling Competition winning algorithm have been used applying the Evaluation Suite to recall the accuracy of the classification. Results are given in the following table.

Subject A	Subject B	Average
97,00%	96,00%	96,50%

2. Testing the performance of SVM training and classification functions: Pre-processed data of the BCI III P300 Spelling Competition has been used. Evaluation Suite SVM training and classification functions have been applied obtaining the following results.

Subject A	Subject B	Average
97,00%	98,00%	97,50%

3. Testing the performance of the entire system. The dataset of the BCI Competition III has been used. Evaluation Suite's pre-processing, SVM training and classification functions have been applied obtaining the following results.

Subject A	Subject B	Average
97,00%	97,00%	97,00%

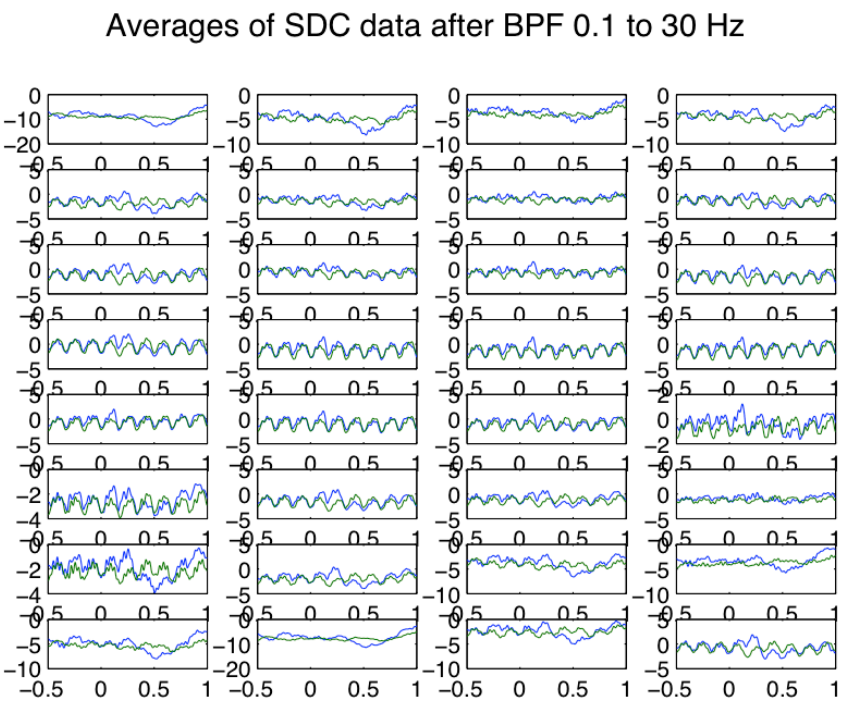
[4] http://www.bbci.de/competition/iii/desc_II.pdf

The winning strategy of the BCI competition III P300 Speller achieves a detection accuracy of 96.5% for no channel selection algorithm. As the detection accuracy of the Evaluation Suite reaches 97%, we assume the complete system has been right reproduced for its usage within the Evaluation Suite. Moreover we have observed that the pre-processing stage could be improved in order to improve the performance level of the system.

2.1.2.3 Performance evaluation on Starlab data set

It is worth first mentioning the data set we are analysing was acquired for 3 subjects. All 3 go through 2 different protocols. The first protocol was done with a speller matrix of 3 x 3 positions, where each position was filled with a device or action icon. The idea here was to use the p300 for device control. In the second protocol we have a collection of 10 images. These ten images are shown at 200 ms rate in random order for 5 times, what we call 5 different runs. We repeat this session 4 times. The protocols were implemented using the software Presentation from Neuro Behavioural Systems. Presentation is a stimulus delivery and experimental control program for neuroscience. A complete description of the porotocol and the obtained data can be found in D4.3 [69].

The first goal of the analysis was to set up the components of the pre-processing stage (see conclusions of the former section). With 1 of the subjects we have visually inspected the results for 3 different pre-processing configurations. Best results are obtained for BPF 0.1-30 and demeaning as pre-processing and the Rakoto pre-processing, which includes BPFiltering 0.1-10 Hz and decimation by a factor 12, in a second step (see Figure 7).



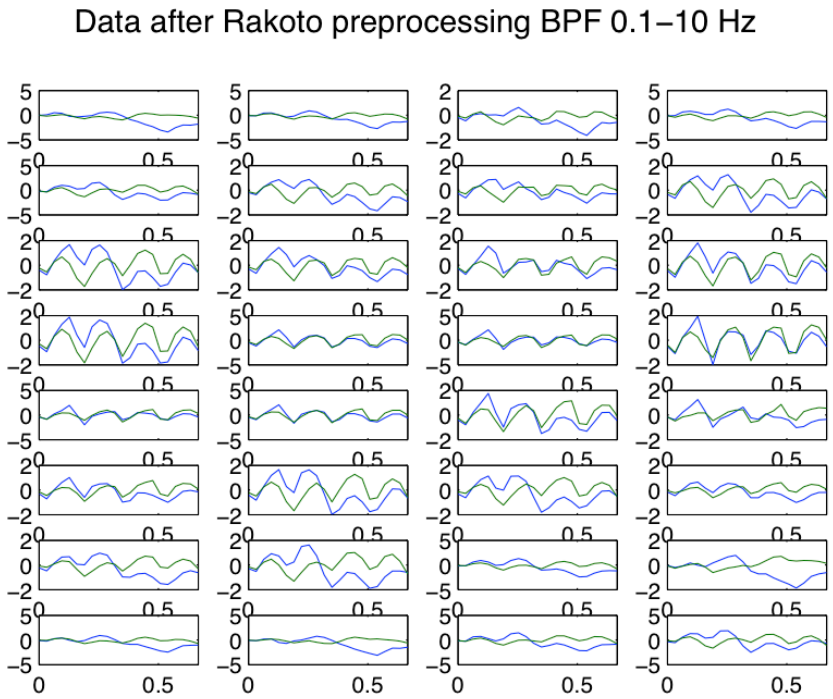


Figure 7. Subject 1, block 3 averaged BPFiltered 0.1-30 Hz (top) and averaged result after BPFiltering 0.1-10 Hz (bottom) of the data acquired under the speller protocol. Averages are taken over epochs of attended (blue) vs unattended (green) stimuli. Chan. 1 to 32 (top-left to bottom-right) are placed following the 10/20 system. Time axis is in sec, and 0 is placed in stimuli onset.

The stimulus onset label seems to be delayed. This would indicate a bad synchronization of presentation with the Biosemi recording. This supposes a shift in the P300 peak (not appearing at 300ms but a bit earlier. This bad synchronization needs to be confirmed by further analysis. This would advocate for not using presentation as already advised by Brendan Allison in a personal communication. Fortunately the delay is constant (around 90 ms) and no jitter could be observed. The corresponding delay can be for instance observed in channel 28 in Figure 7 (supposing this is a P300 EP).

When analysing the set of 3 subjects we could observe some of the channels with something similar to P300 waves in 2 (subjects 1 and 2) out of the 3 subjects. A sanity check on subject 1 block 2 done by permuting the epochs (so that GT does not correspond anymore to the epochs) show a random signal and no difference between attended and unattended. P300 similar waves can be found in channels 1-4, and 27-32. For other blocks of the same subject this could be only clearly observed in channel 28 (see Figure 7 bottom), although for the formerly mentioned channels some differences between attended and unattended can be observed as well.

Apart from this good news one strange thing is the fact that the P300 appears in frontal channels, e.g. indices 1-4, 29-30, as well as in channels where it is supposed to appear like 31-32 (see Figure 1). It would be important to do a channel selection since a lot of channels seem to be just noisy. This could be done a priori by taking a subset of the data set and observing at averages over epochs and blocks. For the third subject some differences between attended and unattended averages could be observed Figure 7, i.e. some channels

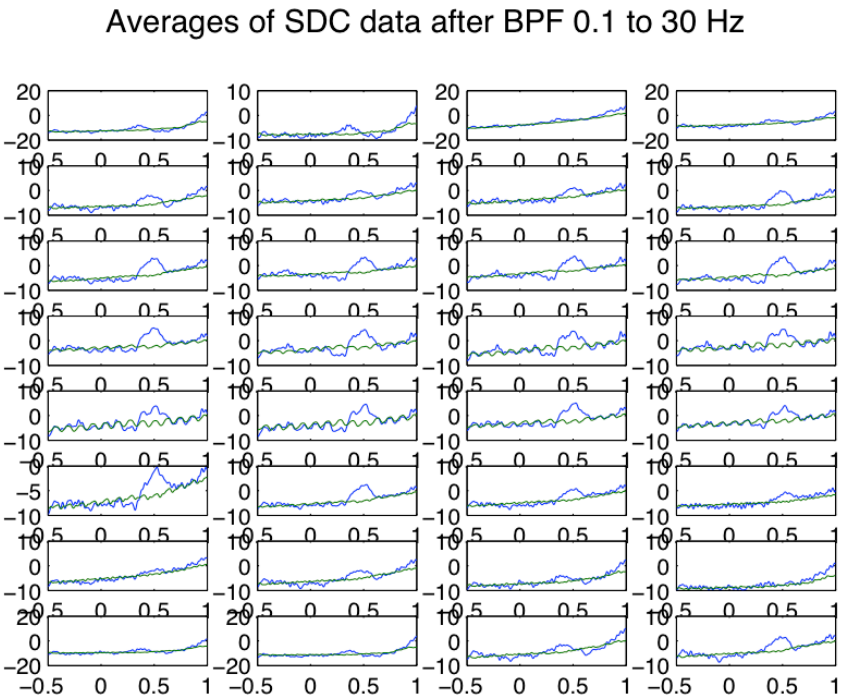
like 25-28 present a peak at around 200 ms that breaks the sine form to be observed for unattended stimuli. This difference is minimal in the sanity check visualization.

As a conclusion of the visual inspection we can state that it is worth testing performance evaluation with subject 1 and 2, although not great results can be expected. For the results to be reasonable a manual channel selection of 1-6, and 25-32 should be conducted. For subject 3 should work even worse.

For the images protocol subject 1 presents some P300-like waves in almost all channels (see Figure 8). Although the P300 wave is not qualitatively good, and it is placed around 450ms, its correspondence is confirmed by the randomized visualization undertaken for sanity check. For subject 2 in this protocol the data does not seem to present any P300 peak, except maybe being very generous on channels 9, 13, 31, and 32. For subject 3 the situation is similar to the results obtained for the speller protocol. In general the obtained averages are again sinusoidal. However some of the channels, namely 13-16, offer differences between attended and unattended stimuli in all 3 analysed blocks. Again the visual inspection for the images protocol advises to undertake a channel selection previous to the performance evaluation. Therefore we undertook following channel selection. The goodness of this channel selection can be confirmed by visual inspection (see Figure 9):

subject	speller	images
1	1-5, 26-32	1-2, 7-12, 21-24, 30-32
2	1-3, 28-31	13-14, 19, 21-24, 31, 32
3	1-3, 26-31	1-5, 7-12, 14-15, 20, 23, 24, 30, 32

Table 1. Selected channels for further analysis of data from both protocols.



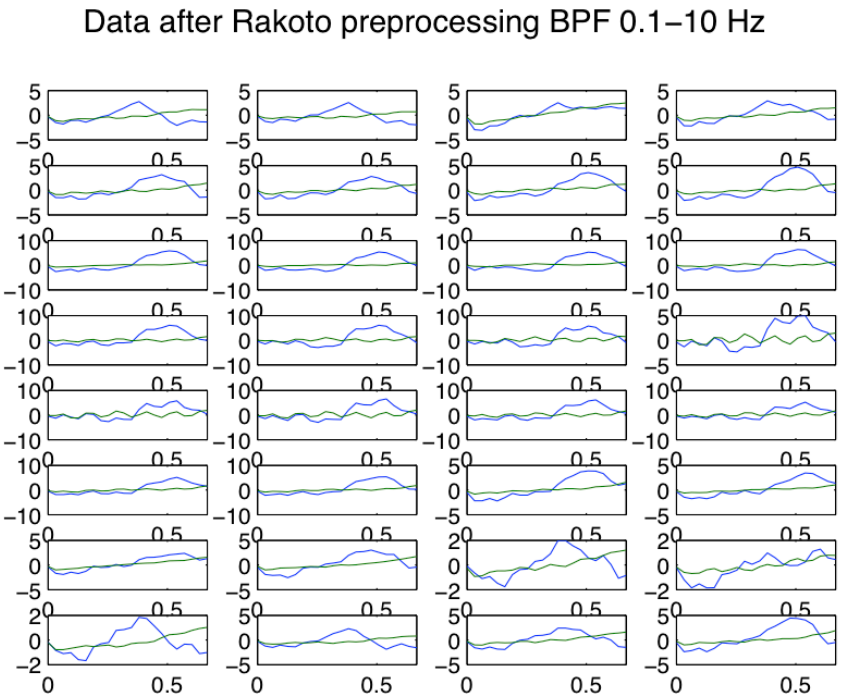
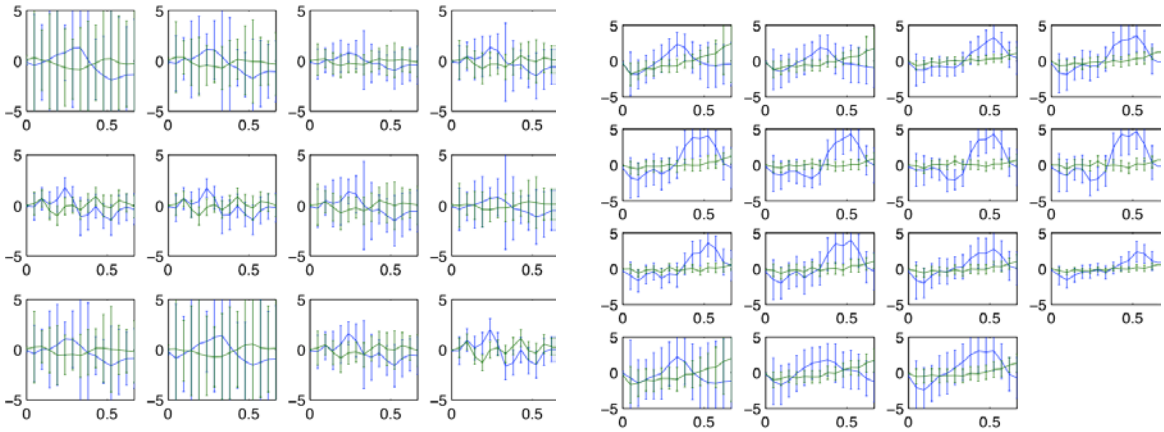


Figure 8. Subject 1, block 1 averaged BPFiltered 0.1-30 Hz (top) and averaged result after BPFiltering 0.1-10 Hz (bottom) of the data acquired under the images protocol. Averages are taken over epochs of attended (blue) vs unattended (green) stimuli. Chan. 1 to 32 (top-left to right-bottom) are placed following convention in 10/20 system.



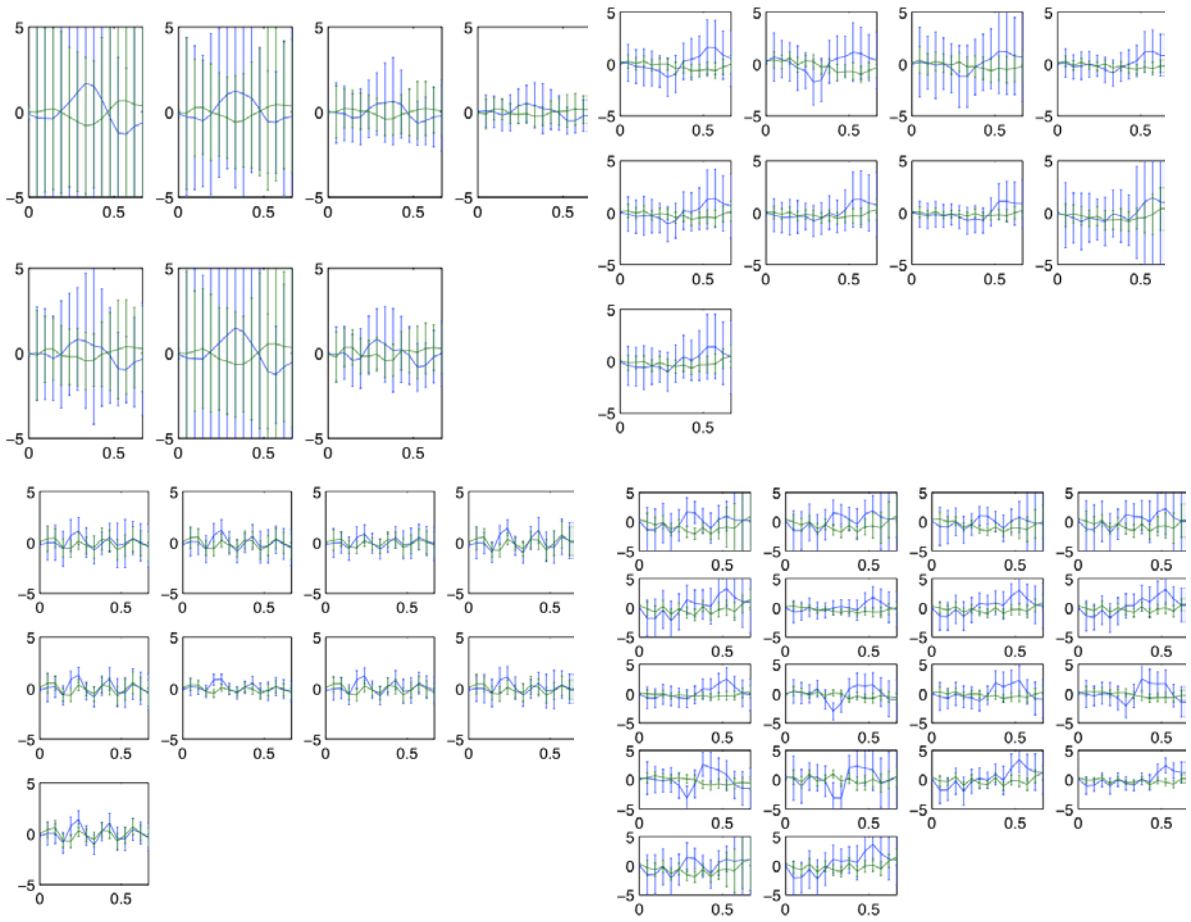


Figure 9. Selected channels averages over blocks with variances (as error bars) for the 3 subjects (top, middle, bottom) in the two protocols "speller" (left) and "images" (right).

The system we have presents 9 classes in the speller protocol, and 10 classes in the images one. Therefore we have changed the methodology as described in Sec. 2.1.2.1 to work with 10 partitions of the data and not with 17, which were selected for 35 classes in the BCI competition data set II. Such a configuration implies that we work with a classifier ensemble of 10 SVMs and not the original 17. We have played with 2 concrete variables in the methodology, namely the channels to select among the initial selected subset given in Table 1, and the classes to be assigned to each member of the ensemble. We have computed the classification rate, i.e. number of right classified examples per total number of examples, for 4 different configurations of the system as described in the following paragraphs. It is worth mentioning that the classification rate is computed over the assignment problem of 9 classes for the speller and 10 for the images protocol and not the attended versus unattended stimuli classification. So the following figures are those of the performance of an eventual application of the methodology.

We compute first the classification rate for the overall system working in the selected feature space as depicted in Figure 9. Here the ensemble is built with the following GT conditions for the partitions, where the indices correspond to classes, e.g. 1 is the first position in the speller matrix or the first image in the sequence, 7 is the 7th matrix position:

speller → [1 2;2 3;3 4;4 1;5 6;6 7;7 8;8 9;9 5]; images → [1 2;2 3;3 4;4 5;5 1;6 7;7 8;8 9;9 10;10 6]. Results are given in Table 2.

Protocol	Subject	1	2	3	Average
speller	Train	100,00%	97,22%	100,00%	99,07%
	Test	69,44%	25,00%	63,89%	52,78%
images	Train	100,00%	92,50%	100,00%	98,00%
	Test	92,50%	52,50%	90,00%	78,33%

Table 2. Classification rates for test 1 (see text for details).

Table 3 gives the results when we have realized a further channel selection for leaving 4 channels, i.e. the number of Enobio's available channels: subject 1 → 5:8; subject 2 → 1,2,4,7; and subject 3 → 12,13,15,16. GT conditions are the same as in the former point.

Protocol	Subject	1	2	3	Average
speller	Train	97,22%	52,78%	88,89%	79,63%
	Test	77,78%	27,78%	61,11%	55,56%
images	Train	100,00%	75,00%	100,00%	91,67%
	Test	85,00%	47,50%	80,00%	70,83%

Table 3. Classification rates for test 2 (see text for details).

We have realized no channel selection and changed GT conditions to have one SVM for each GT class, e.g. speller → [1;2;3;4;5;6;7;8;9]. Results appear in Table 4.

Protocol	Subject	1	2	3	Average
speller	Train	100,00%	91,67%	91,67%	94,45%
	Test	58,33%	19,44%	58,33%	45,37%
images	Train	100,00%	90,00%	100,00%	96,67%
	Test	92,50%	50,00%	90,00%	77,50%

Table 4. Classification rates for test 2 (see text for details).

Lastly in Table 5 we give the results obtained when we operate with the channel selection proposed by Rakotomamonjy and keeping the original GT conditions to have one SVM for each 2 GT class, as in results given in Table 2.

Protocol	Subject	1	2	3	Average
speller	Train	100,00%	97,22%	94,44%	97,22%
	Test	69,44%	25,00%	69,44%	54,63%
images	Train	100,00%	82,50%	%	60,83%
	Test	92,50%	42,50%	%	45,00%

Table 5. Classification rates for test 2 (see text for details).

Performance is good for the images protocol and not so good for the speller one. In both cases there is room for methodological improvements though. In case of the speller data the algorithm seems to overfit. This could be solved by changing the SVM type to linear given the high dimensionality of the data. For the moment we will work further with the images

protocol data. Moreover we take as ground system configuration the one used for the results given in Table 2.

One aspect interesting to analyse is the performance with respect to the decision rate. For this we have tested how the system behaves when averaging a variable number of runs, i.e. a complete sequence of 10 images shown. The evaluation of the fusion performance for the3 subjects is depicted in

Figure 10. As it can be observed the performance seems to be reasonable from 3 runs averaging on. All achieved performances even for 1 run, are far over the random level (0.1, since there are 10 images).

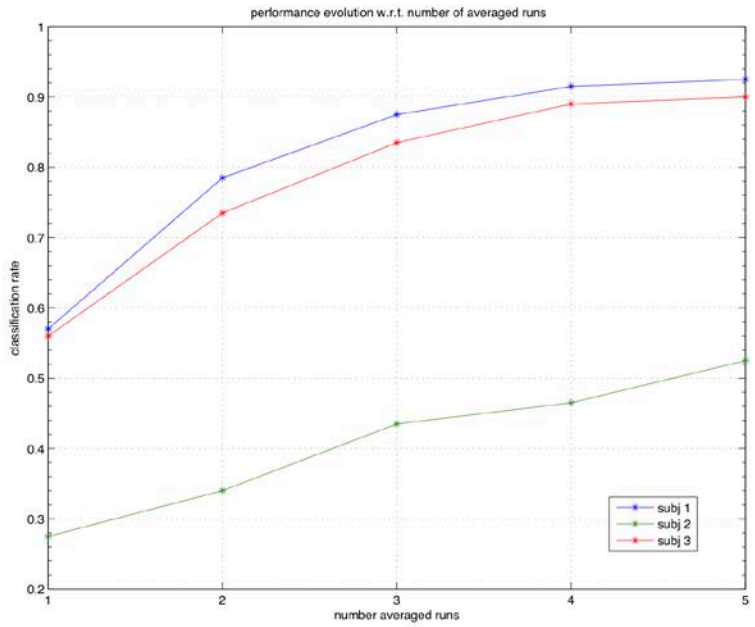


Figure 10. Performance evolution with respect to the number of fused runs in the images protocol for the 3 subjects (see legend).

The evaluation of performance for a variable number of runs entering the double averaging stage are reasonable for 2 of the subjects from averaging 3 runs on. This means the minimal transfer rate of the P300 application for image browsing is 300 ms, i.e. 10 ms for 10 images in 3 runs.

One aspect interesting for the applicability of the framework with Enobio is this of the best channel set to be used in further experiments. With Enobio being limited to 4 channels we looked for the best 4-channel set in the last experiment. We computed therefore the classification rate for each possible 4 combination among the channels previously selected and given in Table 1. We have conducted this computation just for the images protocol. We have then computed the absolute maximum of this classification rate for the test data set, which delivers the best 4-set for each subject. This is given in the following table.

Protocol	Subject	1	2	3
images	Best channels	AF3-T7-CP1-CP5	Pz-PO3-T8-Cz PO3-CP6-T8-Cz	FC1-C3-P7-PO3 FC1-CP5-PO3-T8 C3-P7-PO3-C4 C3-P7-PO3-Cz CP1-P7-PO3-Cz P7-P3-PO3-C4 P7-P3-PO3-T8 P7-PO3-C4-T8
	Test perf.	100.00%	55.00%	90.00%

Table 6. Performance in terms of classification rate over the test data set of the best set of 4 channels for each subject.

As it can be observed, performance improves when selecting the right subset of channels (for subjects 1 and 2), even achieving perfect classification over the test set for one of the subjects. The channel selection is therefore a very important stage.

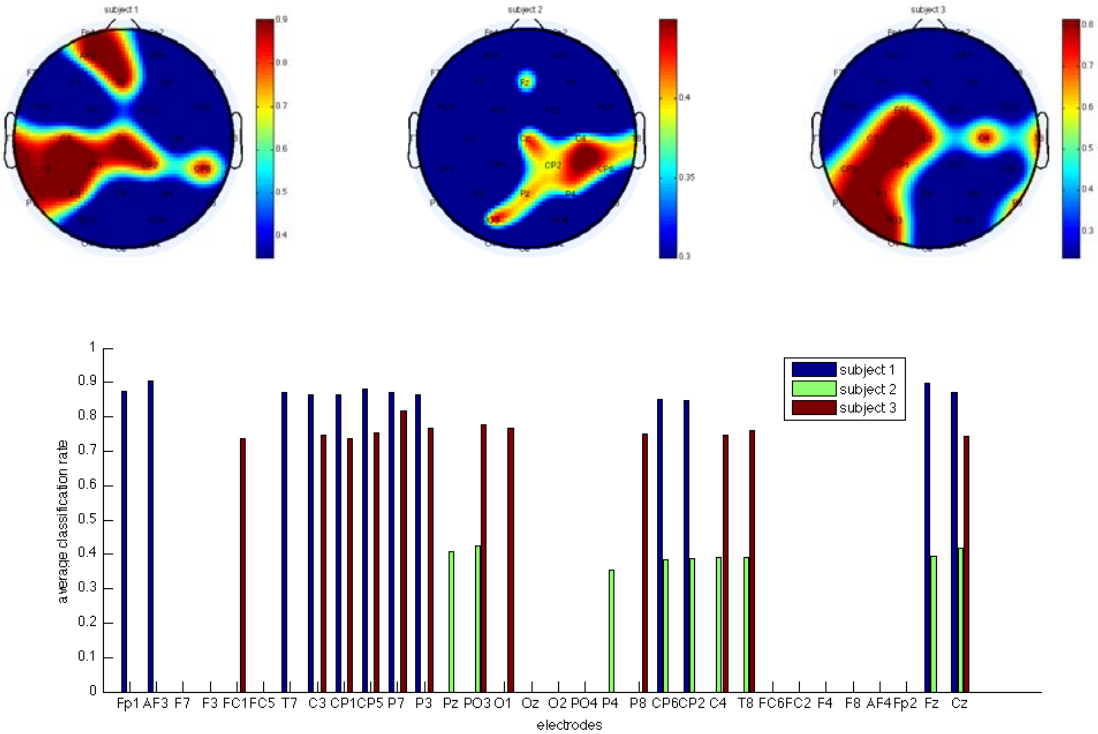


Figure 11. Average classification rate for each channel. Scalp maps subject 1 (right), 2, and 3.

In order to avoid leaving out any important channel in the selection of these 4 channels we have computed the average of the classification rate for each channel as well. For this we compute the average over the results obtained for all the 4-sets where each channel appears. We depict this average classification rate in form of bar plots and scalp maps in Figure 11.

2.2 BNCI SSVEP functionalities in the BNCI Evaluation Suite

SSVEP is an Evoked Potential that consists of a periodic component of the same frequency as a flickering light source the subject is looking at, as well as of a number of harmonic frequencies. SSVEP's are commonly used in BCI applications due to their excellent signal to noise ratio, they are relatively immune to artefacts and provide a high information transfer rate.

This section describes the SSVEP processing and detection methods and algorithms implemented within AsTeRICS WP4 and integrated in the BNCI Evaluation Suite. Enobio SSVEP Demo Platform, which constitutes a Matlab based application especially designed as a first approach to evaluate the performance of the implemented methods, is also presented.

2.2.1 SSVEP integrated functions

2.2.1.1 Signal decomposition

SSVEP's can be recorded from the scalp as a nearly sinusoidal oscillatory waveform having the same fundamental frequency as the stimulus and often including some higher harmonics. A steady-state evoked potential is a repetitive evoked potential whose constituent discrete frequency components remain constant in amplitude and phase over an infinite time period. SSVEP response for an electrode 'i' can therefore be defined as follows:

$$SSVEP_i(t) = \sum_{K=1}^{Nh} a_{k,i} \sin(2\pi Kft + \theta_{k,i})$$

where:

- t is the time.
- Nh is the number of harmonics.
- K is the harmonic number.
- i is the electrode number.
- f is the stimulation frequency.
- $a_{k,i}$ is the amplitude of the sinusoid at the harmonic K for the electrode i.
- Theta is the phase of the sinusoid at the harmonic K for the electrode i.

The brain background activity for the electrode i, which will be denoted as BGi(t), involves all the other processes that are taking part in the brain not related to the SSVEP response. The measurement can also record external disturbances that are added to the measurement as several interferences, for example, from the power line. For the electrode i this is grouped in the noise term, Ni(t).

A linear model of the signal measured in the electrode 'i' can be expressed as follows:

$$Y_i(t) = SSVEP_i(t) + BG_i(t) + N_i(t)$$

2.2.1.2 Matrix implementation

For a measured time window the contribution of a single harmonic in the SSVEP response can be expressed as follows:

$$SSVEP(s) = \sum_{s=1}^{Ns} a_{fs} \sin(2\pi(s/fs)\, fh + \theta_{fs})$$

where:

- s are the samples.
- Ns is the number of samples of the time window.
- fs is the sampling frequency.
- a_{fs} is the amplitude of the harmonic response.
- fh is the harmonic frequency under evaluation being the harmonic number multiplied by the stimulation frequency, i.e. kf.

The vector space, where the contribution of the harmonic lies on, can be defined as follows:

$$X = \left[\sum_{s=1}^{Ns} \sin(2\pi(s/fs)\, fh), \sum_{s=1}^{Ns} \cos(2\pi(s/fs)\, fh) \right]$$

X=[X1, X2] is formed by two linear independent vectors (none of them can be written as a linear combination of the other) and therefore form the base of a vector space. The measured signal Y is going to have components in the defined vector space and others that do not belong to it corresponding to the external noise and background activity.

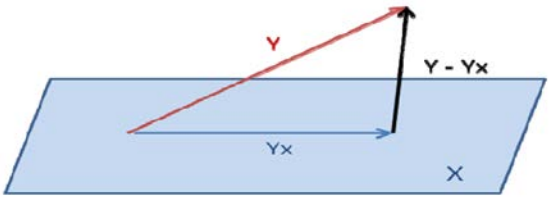


Figure 12. Projection space for the measured signal Y.

The projection matrix of the measured time window Y over N electrodes into the vector space X can be calculated as follows:

$$P = X\,(X^T X)^{-1} X^T$$

It is possible now to determine which components of the measured signal belong to the vector space X (SSVEP) and which do not correspond (nuisance signals and noise).

$$Y_x = PY$$

$$Y_y = Y - Y_x$$

2.2.1.3 Spatial Filter Computation

It is assumed that SSVEP is not only detected in a single electrode and so it is produced in a region of the scalp that involves a certain number of electrodes. As the model generated is linear the goal will be to combine the signal measured in each electrode to obtain an optimal SSVEP response detection, the maximization method described in [70] has been implemented and integrated in the BNCI Evaluation Suite.

Given a signal (Y) of Ns samples recorded through Ne electrodes as a matrix Ne x Ns, the weight factor vector (w) of dimension Nex1 has to be calculated in order to obtain the optimal electrode combination signal, delivering an output vector S whose dimension is 1xNs.

$$S = \sum_{i=1}^{Ne} Y_i w_i$$

$$S = Yw$$

In order to increase the signal to noise ratio the energy of the measurement shall be maximized (Y) while the energy of the background brain activity and noise shall be minimized, obtaining the following maximization problem:

$$max_w \| \frac{Yw}{Bw} \|^2 = max_w \left(\frac{w^T Y^T Yw}{w^T B^T Bw} \right)$$

The equation above is called Rayleigh Quotient. This problem is solved obtaining the generalized eigenvectors. The eigenvectors obtained are orthogonal and all of them will be a different solution of the maximization problem.

$$[W, \lambda] = eigen(Y^T Y, B^T B)$$

All the Eigenvalues are larger than or equal to one, and the portion larger than one show how much larger the energy in the SSVEP frequencies is in comparison to the non SSVEP frequencies. The Eigenvector corresponding to the largest Eigenvalues is the one that delivers the largest signal to noise ratio.

2.2.1.4 SSVEP training

SSVEPs are a subject and stimulation frequency dependent phenomena where:

- The simulation frequency that elicits the largest response depends on the subject.
- The harmonic that elicits the largest response depends on the subject and the stimulation frequency.
- The spatial filter that elicits the largest response depends on the subject and the stimulation frequency.

A training stage that extracts the spatial filter for the harmonic that elicits the largest SSVEP response has proved to be necessary. For this purpose a training signal is needed where stimulation and non-stimulation periods could be extracted.

This training signal shall consist of Nstim number of stimulation periods at the stimulation frequency under evaluation and Nnonstim non stimulation periods. Electrodes not placed in

the occipital area shall be removed as it is assumed that SSVEP only arises in the visual area. For each training signal the performance of each harmonic (K) shall be individually evaluated.

This training process delivers the performance of a stimulation frequency at a given harmonic. The stimulation frequencies and the spatial filter associated to be used in a final BCI SSVEP based application shall correspond to the ones that deliver the best performance. The first step is to extract the Nstim spatial filters corresponding to each of the stimulation periods as described in the former section (see Figure 13).

Once that the spatial filters are extracted their performance is evaluated over the same training signal. After the training signal spatial filtering the energy in both the stimulation and non-stimulation regions is extracted. To evaluate the detection performance the area under the ROC curve is calculated where the stimulation periods are marked as the positive class and the non-stimulation periods as the negative class. The largest AUC delivers the best performance for the harmonic under evaluation. The spatial filter and the harmonic associated that elicits the largest AUC shall be taken as best harmonic and spatial filter to be used in the future SSVEP detection for the subject at that stimulation frequency.

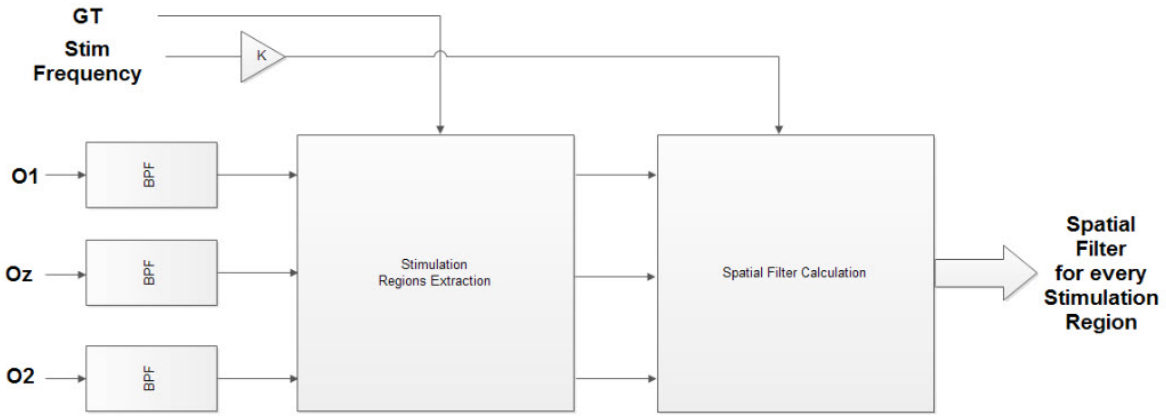


Figure 13. Computation of Spatial Filters for each stimulation period in the training sequence.

2.2.2 Enobio SSVEP demo platform

A Matlab/Simulink application has been developed to control the stimulation source, carry out the recording protocol and evaluate the performance of the detection algorithms implemented. This model makes use of the SSVEP Evaluation Suite functions described in the former section.

The application takes advantage of the TCP-IP data streaming capabilities of Enobio Software V2.0. Enobio SSVEP Demo Platform connects to the TCP/IP port of the Enobio Software acquiring, monitoring and analysing the EEG data streamed in real-time. TCP communication provides the added advantage of running Enobio SSVEP Demo Platform on the same computer as Enobio Software or remotely.

The User Interface (see Figure 14 and Figure 15) has been designed to provide a user-friendly navigation through all the possible options the GUI offers. Enobio SSVEP MI User Interface has been conceived as an integrated user interface where it is possible to reach every option the system offers over a single panel.

2.2.2.1 Panel Control and Configuration

The application controls and configures the four external stimulation panels, developed by FH Technikum Wien, in charge of eliciting the SSVEP response. The following parameters can be individually customized at each stimulation panel:

- Frequency: Number of On-Off periods repeated per second.
- Duty cycle: Ratio of the duration of light on sub-period to the total period.
- Intensity: Luminance of the on sub-period
- Colour: Additive combination of red green and blue.

Once that the stimulation panels have been configured it is possible to manually command a stimulation start/stop.

2.2.2.2 Manual Recording

It is possible to manually record the EEG measurement acquired along with the stimulation trigger. The data is recorded in a plain ASCII file where the corresponding Enobio channels are stored in the first four first columns and the stimulation trigger in the fifth column. In non-stimulation periods the data trigger is set to zero and during the stimulation periods the data trigger is set to the stimulation frequency in Hertz of the Panel #1.

2.2.2.3 Stimulation Training Protocol

The Enobio SSVEP Demo Platform automatically runs a full training protocol for a given number of stimulation frequencies, in this case 12, 14, 16, 18, 20 and 22 Hz. It records a training measurement for each stimulation frequency where the SSVEP recall parameters shall be calculated. The number of stimulation periods is customized by the user.

The protocol fulfils the following requirements:

- In order to avoid the effect of processing transient times a transient time equal or larger than 5 seconds shall be recorded at the beginning of each measurement.
- The stimulation period duration is randomly chosen between 3 and 4 seconds to be considered long enough to evaluate good detection while not uncomfortable for the subject.
- The non-stimulation periods shall be randomly chosen between 5 and 8 seconds to be considered time enough for having a sufficient resting time between stimulations and a sufficient period for the non-stimulation detection.
- The subject shall know beforehand when the stimulation period starts to avoid blinking and focus his/her attention at the stimulation panels. For this purpose a beep sound will be played one second before the stimulation period starts.

2.2.2.4 SSVEP Frequency Train

The Enobio SSVEP Demo Platform configures the detection for each of the stimulation frequencies defined in each stimulation panel. For each stimulation panel it calculates the spatial filter and its best associated harmonic over a previously recorded training measurement, as explained in the former section.

2.2.2.5 Signal Monitoring

The Enobio SSVEP Demo Platform monitors in real time the signal acquired in each electrode along with the stimulation trigger in real time as seen in Figure 15.

2.2.2.6 Real Time Feature Analysis

The Enobio SSVEP MI application allows real time feature visualization to evaluate the response of the evoked potential elicited. This is done through the GUI given in Figure 14. The feature analysis has been designed to work with the default set up, four panels connected and each one flickering at a different frequency. A training signal for each of the frequencies shall be previously recorded in order to obtain the spatial filter and harmonic that elicits the largest response. Once the setup has been configured the features the application calculates and represents for each panel configuration are:

- Power at the frequency that elicits the largest response in the training signal
- FFT Coefficients at the frequency that elicits the largest response in the training signal calculated over a 5-second buffered signal with a 100ms shift.
- PSD of the last two seconds acquired.

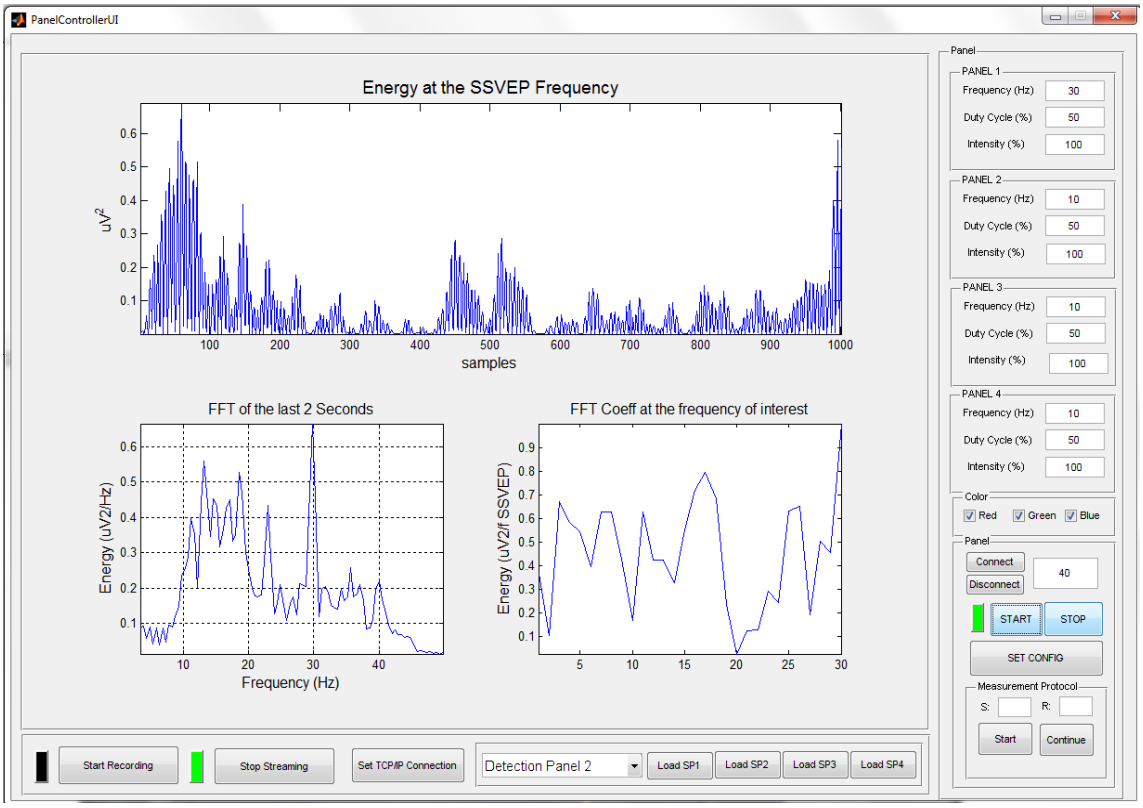


Figure 14. GUI for real-time feature analysis.

2.2.2.7 SSVEP Detection Demo

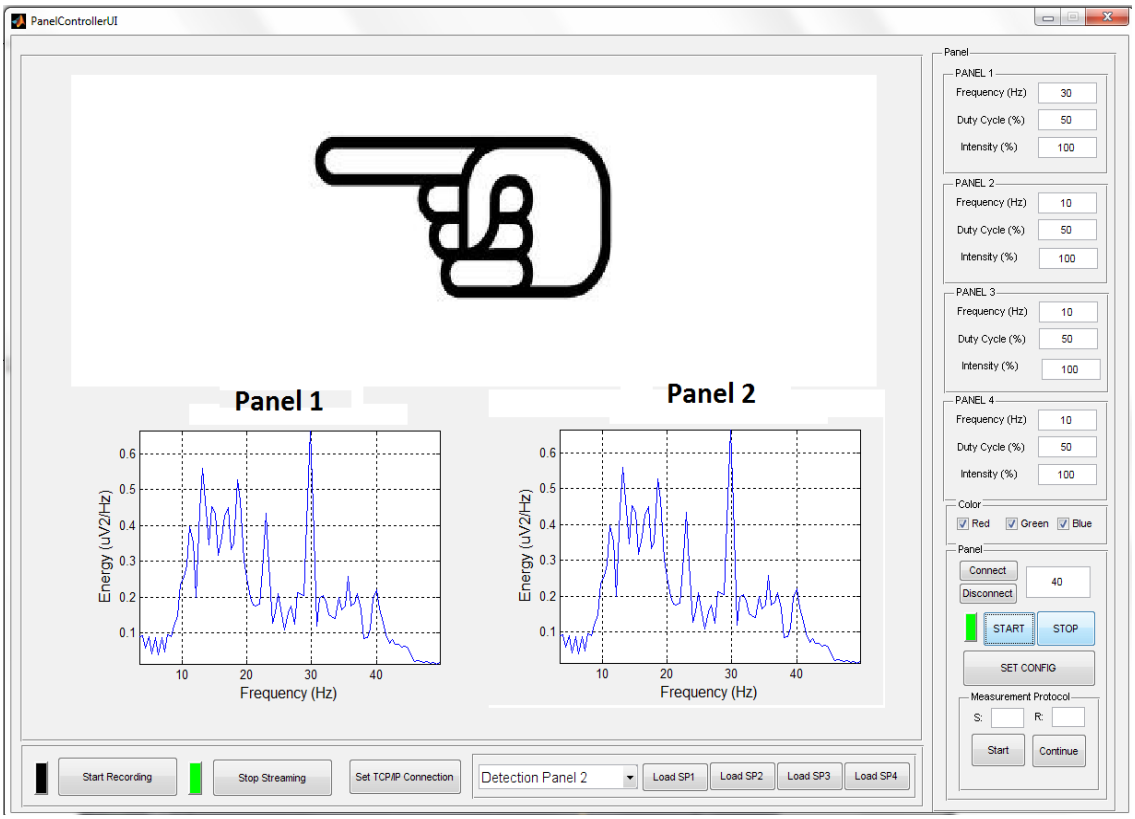


Figure 15. GUI for the SSVEP Detection Demo.

A demo with two stimulation panels has been implemented within Enobio SSVEP Demo Platform (see GUI in Figure 15). This demo assumes that the stimulation panel 1 is placed on the left of the screen, where currently the application is currently running, and the stimulation panel 2 is placed to the right. Panels 3 and 4 are ignored. Their corresponding spatial filter and best harmonic are previously calculated over previously recorded training signals.

After a stimulation the ratio of the FFT coefficients at its best harmonic frequency during the stimulation period versus the last non stimulation period are calculated. The panel that delivers the largest ratio is chosen as the detected one. If panel 1 was detected a hand with a finger pointing to the left is displayed, otherwise a hand with a finger pointing to the right is displayed.

2.3 AsTeRICS works for standard BNCI toolkits

2.3.1 Integration with OpenVIBE

Next to BCI2000, OpenVibe by INRIA is one of the leading BCI frameworks today. OpenVibe is open to the research community and to BCI enthusiasts wordwide, and constantly growing since its release in 2008. A number of biosignal acquisition devices are already supported.

The design concept of OpenVibe is similar to the ACS: graphical arrangement of plugins and their interconnection allows the creation of very complex signal processing chains in an intuitive way. The OpenVibe framework provides plugins and demo scenarios for all major

BCI paradigms. The included examples show how classifiers for SSVEP, P300 and movement imagination BCI applications can be trained and how they can be used in online BCIs. For further information please refer to the homepage: http://openvibe.inria.fr.

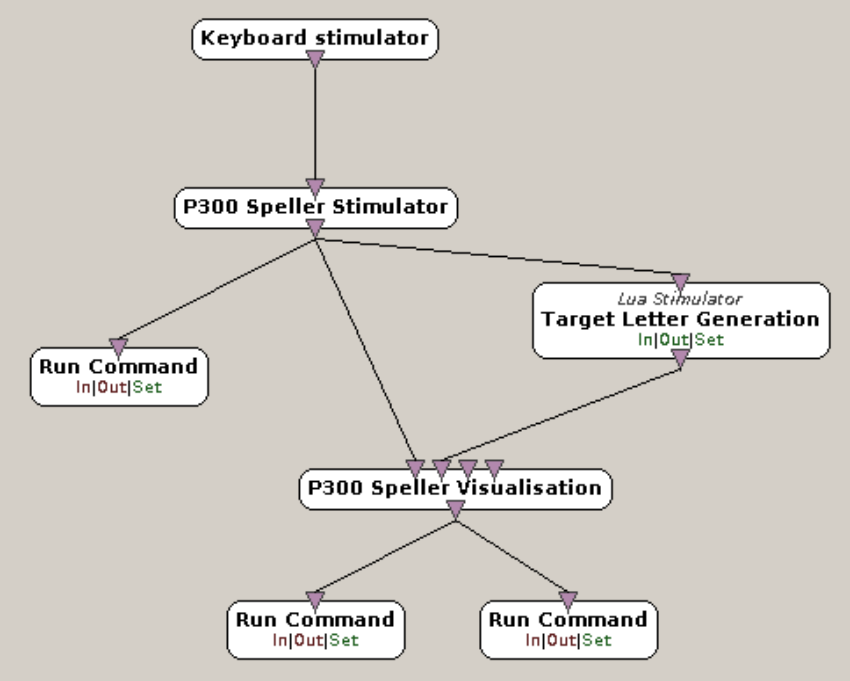


Figure 16: the OpenVibe designer for creation of BCI setups.

The integration of OpenVibe and AsTeRICS consists of two major parts:
The implementation of a device driver to support the Enobio EEG unit in the OpenVibe acquisition server.

The establishment of a data connection from OpenVibe to the ARE, which allows sending BCI information, respectively the results of any BCI classifiers or raw signals, to the ARE.
Thus, the existing resources in both frameworks can be combined, so that the various actuator capabilities for Human-Computer-Interaction and environmental control provided by AsTeRICS can be used via the BCI functions of OpenVibe.

2.3.2 Enobio integration in OpenVIBE

In order to integrate Enobio on the OpenViBE framework, we needed to extend the OpenViBE acquisition server to be able to acquire data from Enobio. Therefore, we implemented a new acquisition driver for Enobio following the OpenViBE acquisition driver interface and making use of the Enobio API.

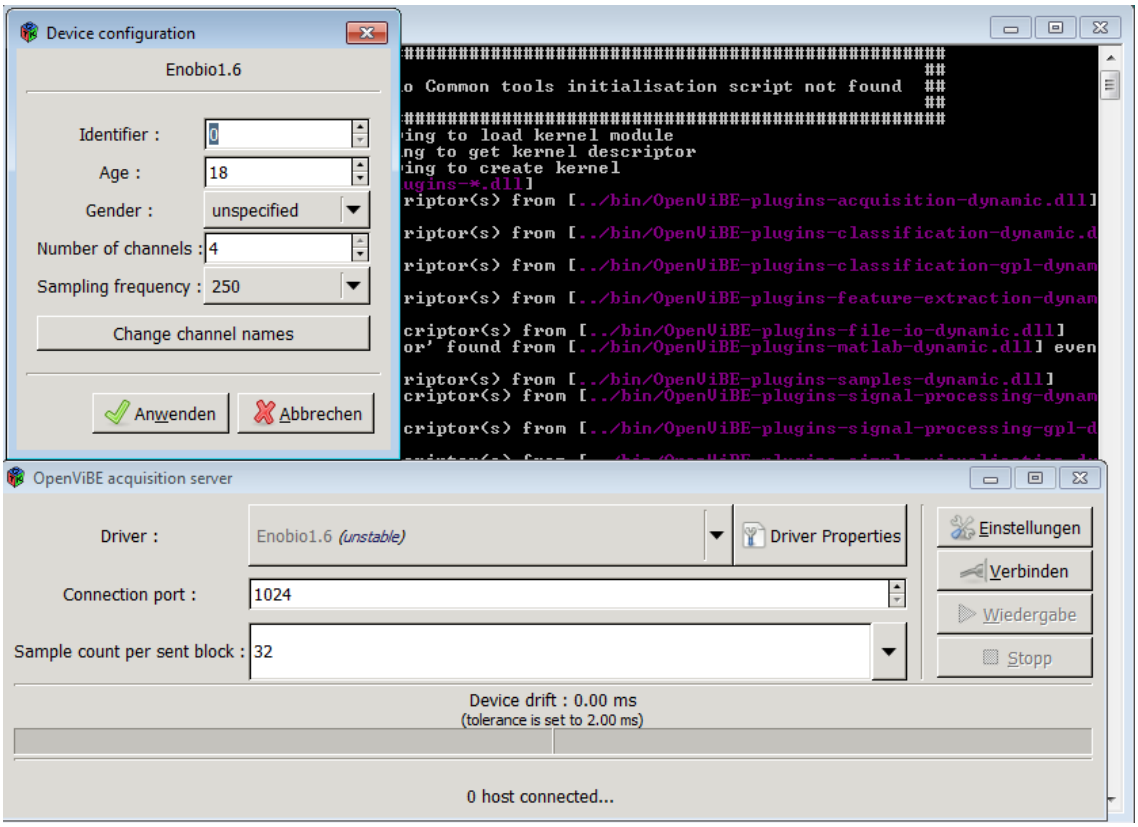


Figure 17: OpenVibe acquisition server, running the Enobio plugin

The OpenViBE acquisition driver interface describes a finite state machine bundled in a class which deals with two kinds of data: the header data and the buffer. The first corresponds to constant data, such as running parameters, identifiers about the data and so on. The second corresponds to all the dynamic data that is modified during the lifetime of the session, such as the samples streamed by the device. In Figure 18, taken from OpenViBE's official webpage, the finite-state machine is described.

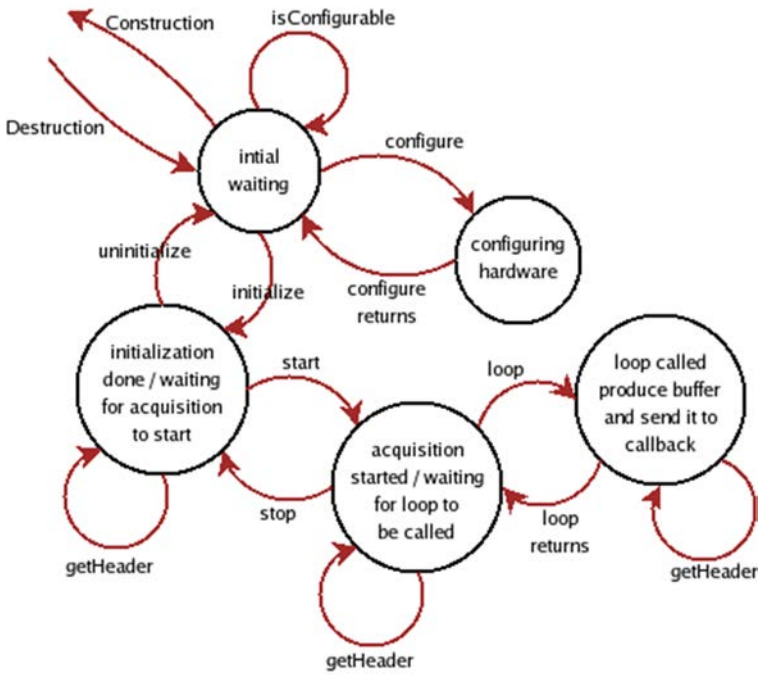


Figure 18. Finite-state machine describing the running process of an OpenViBE acquisition driver. (credited to http://openvibe.inria.fr/)

OpenViBE offers a skeleton-generator application that facilitates the creation of a new acquisition driver. Filling the describing information for the driver (driver and author name, sampling frequency, channel count, …), returns the skeleton implementation files that need to be coded, including the driver class code and the user interface. For the implementation of the acquisition driver we filled the skeleton with all the corresponding calls to the Enobio API and the necessary intermediary data management and flow control routines.

For the implementation, we followed a dual role schema: we implemented a data consumer, in charge of the implementation of the IDataConsumer functionality, and a status consumer, in charge of the implementation of the IDataProducer functionality. The main class of the driver - CdriverEnobioAcquisition - acts as a middleware between the data consumer and the status consumer (data producer).

For the communication with the device, the implemented driver makes use of the direct access to the hardware API. That is, the driver communicates directly with the hardware and receives the data stream, instead of receiving the data through a TCP/IP socket from the usual Enobio client application.

2.3.2.1 Integration of OpenVIBE in the ARE

The strategy of the OpenVibe – AsTeRICS integration was to use a standardized interface / protocol for the connection of the two frameworks, so that the data bridge can be re-used for other purposes and other developers can find an easy start if they want to utilize or improve the work. It became clear that a separation of OpenVibe and the ARE would often make sense, especially when a high performance system is needed for the signal classification in OpenVibe and the AsTeRICS Personal Platform shall mainly be used for the ARE and the connection of actuators. Thus, TCP/UDP was chosen for the implementation of the

connection, and the Open Sound Control protocol[5] was used to structure the data flow. OSC is a widely known format for TCP/UDP connections, often used in audio applications, but completely generic in its capabilities.

Generally in OpenVibe, two concepts are used to organize the live data exchange between plugins:

- Signals (for streaming data, e.g. biosignal data with a specified sampling rate).
- Stimulations (occasional events, e.g. an external visual stimulation marker or the detection of a BCI classification in the online signals).

These two concepts are similar to the signal- and event-channels in AsTeRICS. In fact, OpenVibe signals can be mapped to ARE signals, and OpenVibe stimulations can be mapped to ARE events. The following block diagram shows the intended functionality of the data brigde:

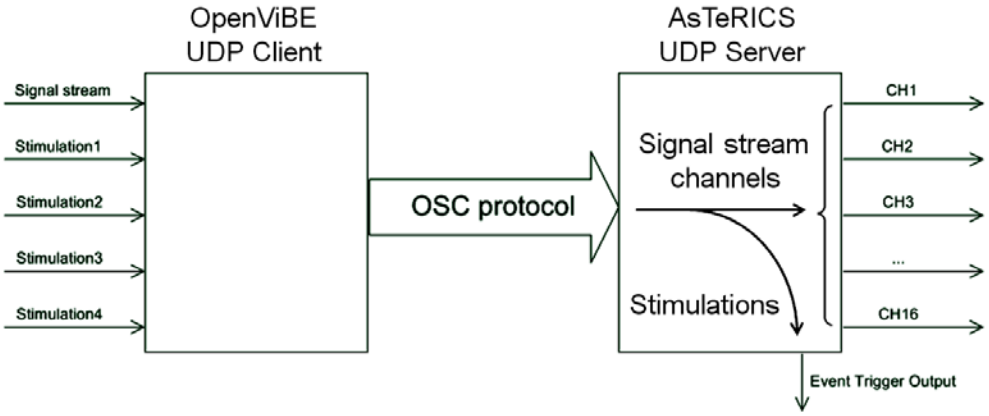


Figure 19. Block diagram of the OpenVibe-to-AsTeRICS connection

The AsTeRICS-plugin (“OpenVibe”) acts as a UDP server and listens on the port specified in the “Port” property. The OpenVibe plugin (“AsTeRICS Connection”) connects to this server port to establish the connection.

The OpenVibe plugin “AsTeRICS Connection”.

The OpenVibe plugin needs to know if the stimulation channels 1 and 2 have to be used for letters (A-Z) or numbers (0-9), or if they are normal stimulations that can be used for further processing. This is defined via the used BCI paradigm-property. The default value is “P300” which then recognizes the stimulations “OVTK_StimulationId_Letter_A” to “OVTK_StimulationId_Letter_Z” and “OVTK_StimulationId_Letter_0” to “OVTK_StimulationId_Letter_9”.

The following screenshot shows the implemented plugin in the OpenVibe designer menu:

[5] http://opensoundcontrol.org/

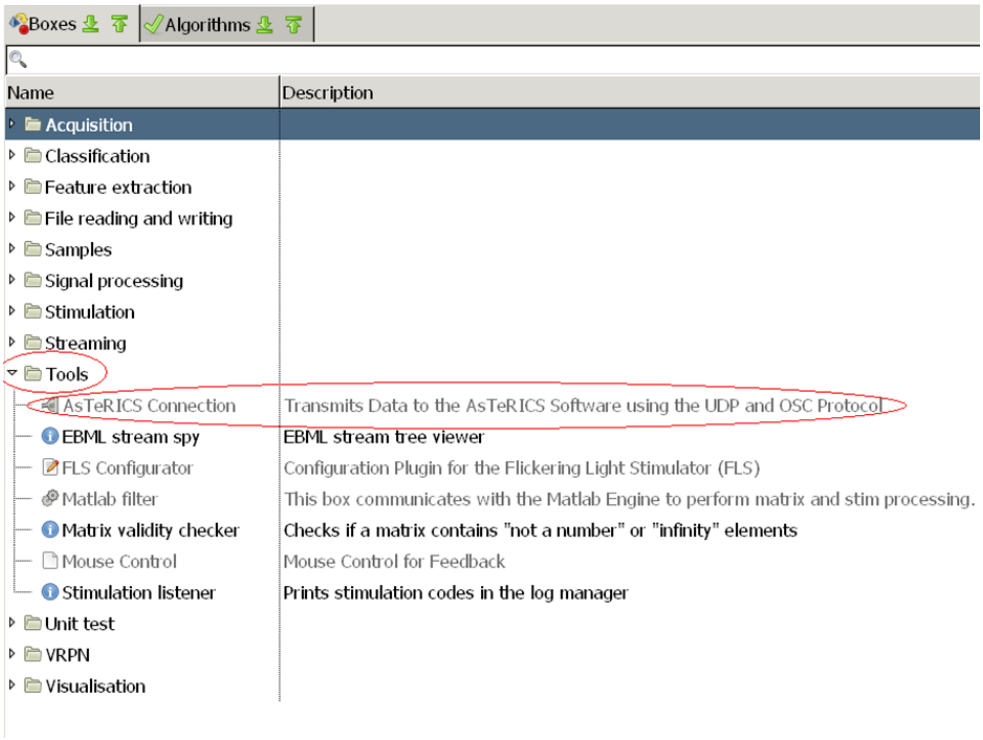


Figure 20. "AsTeRICS Connection" plugin in the OpenVibe designer.

The OpenVibe-plugin has the following properties:

- IP-Address of device on which the AsTeRICS ARE – or to be more detailed, our OSC Server – is running (default value is the loopback address "127.0.0.1")
- Port on which the OSC server is listening (default value is "57110" → the same default value is also used within the AsTeRICS plugin)
- The BCI Paradigm, which is required to let the ARE plugin know how to interpret the stimulations (default value is "P300")

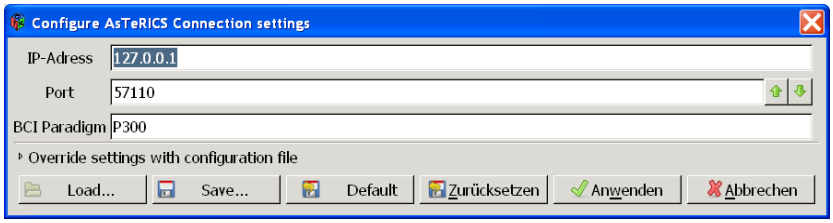


Figure 21. OpenVibe plugin properties

Regarding the implementation, the OpenViBE plugin is structured in 3 main important modules: the initialization, the runtime processing and the uninitialization:

- CBoxAlgorithmAstericsConnection::initialize(): called after pressing start in the OpenViBE designer, initializes all the required buffers for the signal and stimulation inputs, starts Winsock, opens the UDP socket and sets the IP address and port of the listening UDP server

- CBoxAlgorithmAstericsConnection::process(): called every time a new data package is received on one of the five inputs; analyses if the data is either a streamed matrix of signals or a stimulation; prepares a new OSC message (stimulation) or bundle (several channels) and sends it to the given IP address and port of the listening UDP server

- CBoxAlgorithmAstericsConnection::uninitialize(): called after pressing stop in the OpenViBE designer, uninitializes all required buffers, closes the UDP socket

For the OSC data transfer, the C++ OSC library "OSCPack"[6] by Ross Bencina was used. This library provides an OSC C++ class that supports the creation of the sockets and the data transmission via UDP or TCP via the OSC protocol.

The AsTeRICS ARE plugin "OpenVibe":

The ARE plugin features 16 output ports for signals and one event trigger port which can emit 45 different events for stimulations. In case of received signal data, the data packages are split up to the corresponding output channels 1 to 16. Received stimulations produce a corresponding trigger event on the event trigger port. The only property of the AsTeRICS-plugin is the UDP port where the server starts to listen.

Regarding the implementation, the netutil OSC library for Java has been used. Similar to OSCPack on the C++ side, the Netutil library provides a JAVA class for using UDP / TCP sockets and the OSC protocol.

2.3.2.2 The working OpenVIBE – ARE connection

The following screenshots show a working connection between OpenVibe and AsTeRICS. The acquisition server just sends off different sine-waves, which can be seen in the OpenVibe Signal display.

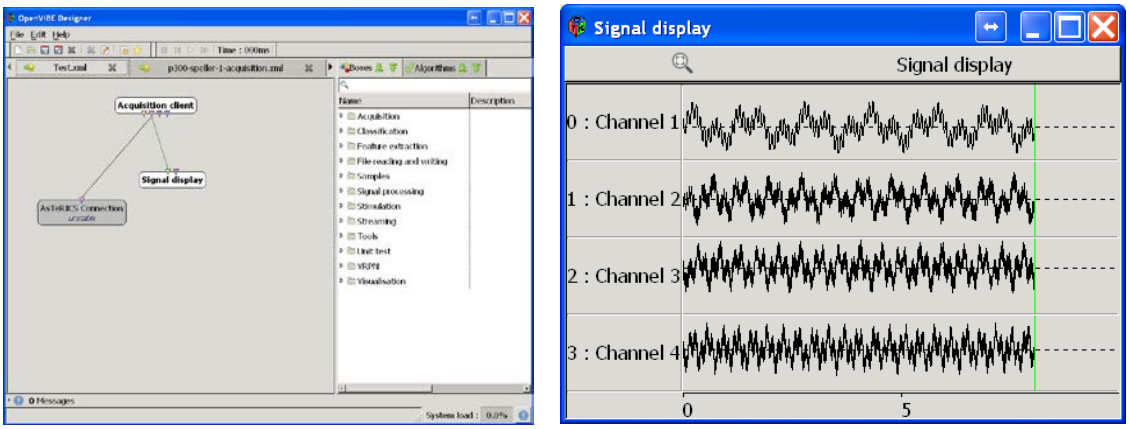


Figure 22. Working connection, OpenVibe side.
Simple Scenario in the OpenVibe Designer (left), transmitted signal in the Signal display (right)

[6] : http://www.rossbencina.com/code/oscpack

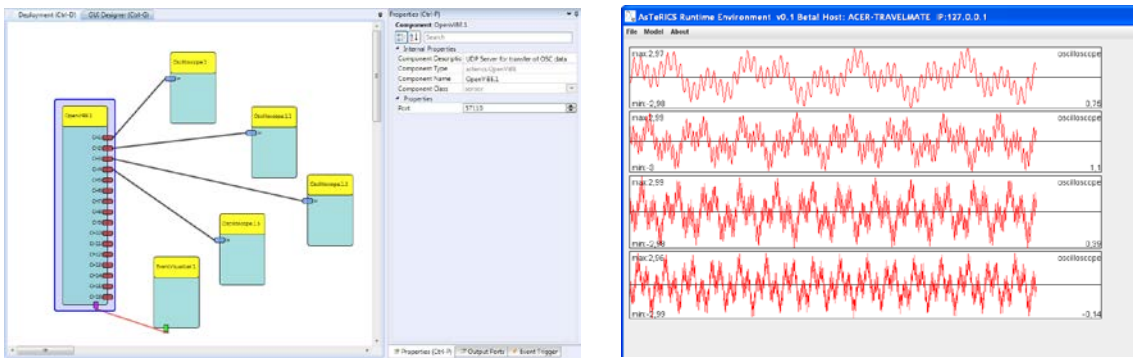


Figure 23. Working connection, AREside.
ACS model to receive UDP data (left), received signals in the Oscilloscope plugin (right)

An SSVEP-BCI controlled CD-player

As a proof of concept, a test setup has been evaluated where OpenVibe scenarios for SSVEP training and online classification were modified to use the AsTeRICS Connection to send classification results of the online BCI to the ARE. The classification results were represented in form of OpenVibe stimulations for the different recognized frequencies, where only two (out of four) were used in this simple test setup.

In the AsTeRICS model, counters were utilized to record the received stimulations, thereby giving an absolute number of the classifications for frequency 1 and frequency 2. A comparator plugin was used to set an adjustable threshold for the functions “Play” and “Stop”. These events were sent to the IRTrans plugin that emitted the respective Infrared Remote codes to control the CD-player.

The setup worked well (recognition rates about 90%) in 2 out of three test subjects without training of the SSVEP procedure. Later, the setup was extended to four frequencies, to add the “next track” and “previous track” functions of the CD player.

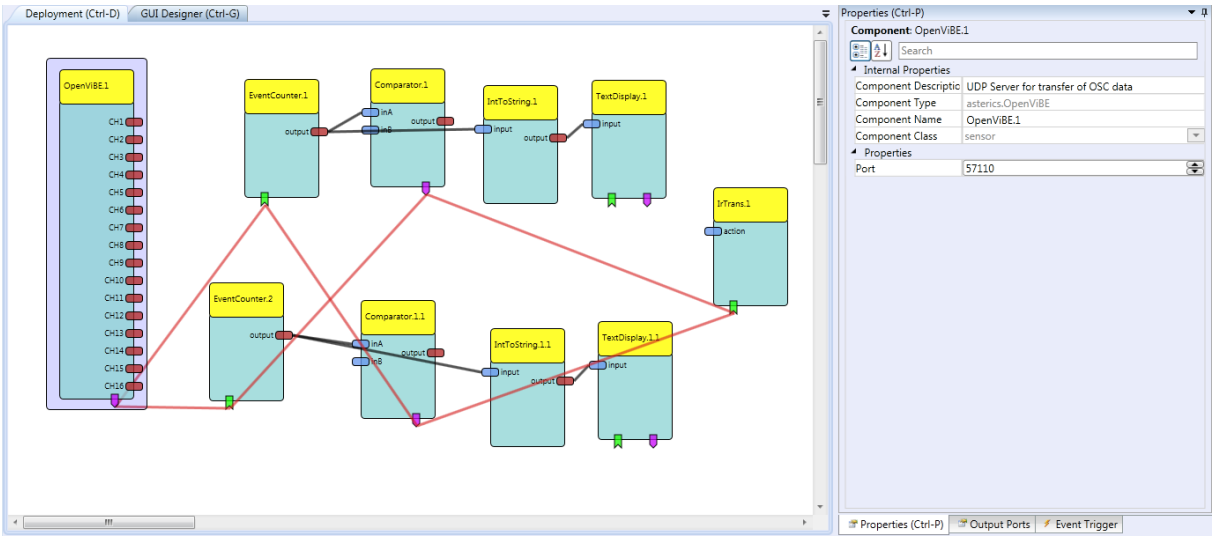


Figure 24. ACS model for a SSVEP-BCI controlled CD-player via the OpenVibe - connection

2.3.3 Proposal for integration of TOBI interfaces

The TOBI project[7] has generated some architecture to ease the connectivity of BNCI system sub-modules, e.g. electrophysiological acquisition devices, classifiers. As proposed by the reviewers the integration of such interfaces in AsTeRICS should be aimed in order to extend the applicability of the AsTeRICS platform. A requirement analysis has been undertaken by Starlab and described herein.

The TOBI project proposes 4 different interfaces for data acquisition, BNCI output transmission, markers and events transmission, and an underlying protocol for communication among sub-modules. In our opinion, the last 2 ones being internal interfaces, the most interesting for AsTeRICS would be the data acquisition, and the BNCI output ones. These two would allow to connect at the input of an application built as an ACS model (data acquisition), and delivering the output of a BNCI sub-module following the TOBI standard to any of the built-in AsTeRICS actuators. Such interfaces succeed in form of the TOBI acquisition interface (TiA), and the TOBI control interface (TiC).

The most straightforward idea for the integration of both interfaces is the implementation of two different ARE signal processing plug-ins, which make use of the software libraries provided with the TiA and the TiC. We analyse in the following the high-level requirements of such plug-ins (see Figure 25).

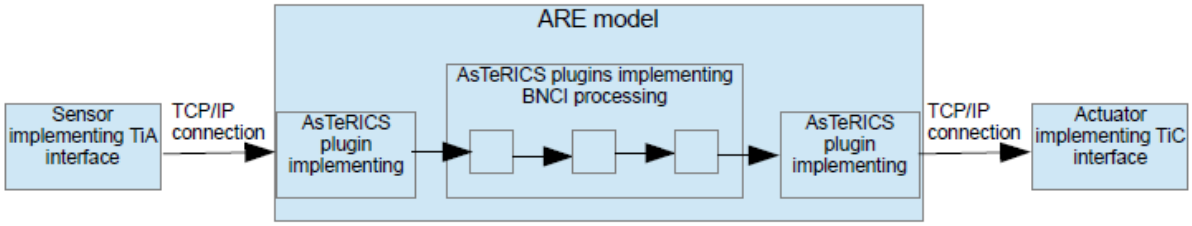


Figure 25. Integration of Sensors and Actuators with a TOBI interface into the AsTeRICS framework.

In the first case the TiA plug-in should implement a TiA client that would connect with the acquisition device server. The purpose of the plugin is to implement the TCP or UDP connection with this server through the specified commands for getting the device data into the AsTeRICS system. Such a solution is similar to the one existent for the OpenVIBE integration. Once done, the plug-in would provide the data in form of the data channels existent in the ACS. One possible problem in this picture is the dynamic nature of the data delivered by the TiA, i.e. the number of channels is known after getting the meta-info through the corresponding TiA command. Given the static data structures existent in the ACS, it would be impossible to implement a generic TiA plug-in. Therefore a separate plug-in for each device to be connected would have to be implemented.

In case of the TiC plug-in this should implement a parser of so-called iC messages. These messages define different fields for the transmission of classifier outputs. One difficulty in this

[7] http://www.tobi-project.org/

case is the lack of definition in the TiC standard of the transport layer. This makes the interface more flexible, but hinders again the implementation of a generic TiC plug-in.

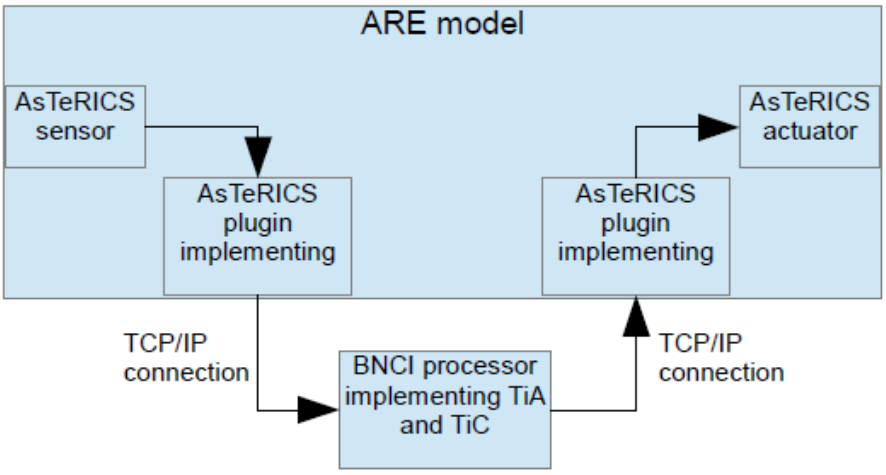


Figure 26. Integration of BNCI module with TOBI interface into the AsTeRICS framework

One further idea would be to integrate in AsTeRICS a BNCI module developed by a third-party using TOBI interfaces (see Figure 26). In this case we should implement the counterparts of the TiA plugin as described above, i.e. a TiA server, and an analogous TiC plugin. Given the requirements, whose high-level description is given above, the implementation effort for such integration is initially assessed as 3 PMs. This estimation should be refined after completing the low-level requirements. The task for implementing such interface plugins was not foreseen when completing the AsTeRICS Description of Work, thus implementation during the AsTeRICS project duration is not feasible.

3 SVM Software Module

The Smart Vision Module (SVM) provides a Computer Vision based sensor for the AsTeRICS Personal Platform. As described in the System Specification and Architecture (D2.1) and in the risk register, the SVM will be implemented in two different versions for the AsTeRICS system prototype-1 (where a remote head tracking system based on webcam image acquisition is desired) and the system prototype-2 (where a head-mounted iris tracking application will be developed).

The main advantage of the remote system is that no devices or sensors have to be mounted on the subject's body, which may increase the comfort for the users, especially in long-term use. The main advantage of the head-mounted system is that eye movements can be measured with high accuracy and that eye-tracking and gaze-estimation applications become possible.

This two-stage approach offers optimal flexibility in the sense of the Assistive Technology Construction Set, as both strategies for camera-based computer interaction will be available at the end of the AsTeRICS project.

In the following, the accomplished SVM implementation for the remote head tracking system will be described in more detail. For the remote SVM, two different algorithms have been evaluated and implemented and are now available as sensor plugins for the ARE:

- A head-tracking approach optimized for speed in a low-performance computing environment, based upon a Haar-Classifier Cascade for face detection and an Optical Flow algorithm for feature tracking.
- A head-tracking approach that offers high face tracking stability, based on Active Shape Models.

Both approaches have advantages and disadvantages. Depending on the current situation and the available computing power, the optimal vision sensor plugin can be chosen.

3.1 Head Tracking via Haar-like Feature Detection and Optical Flow

The "facetrackerLK" sensor plugin of the ARE offers a head-tracking algorithm that calculates the actual movement of the user's nose and chin from a remote camera image, captured by a high-quality web camera. The nose- and chin movement is provided at the plugin's output ports. This data can be used by other plugins for mouse replacement or selection in an on-screen keyboard grid. By computing relative movements of nose and chin via dedicated ARE-plugins, voluntary movements of the chin can be detected and utilized e.g. for selection in an on-screen keyboard scenario or for mouse clicking. The computer vision approach is based upon a combination of a face-detection algorithm (Haar-Classifier Cascade) and a feature-tracking algorithm (Lukas Kanade Optical Flow Tracking), as described in [44]. The plugin has been updated to use VideoInput and Cimg for acquisition and window management. The details of the methods used are not changed w.r.t. the description in D4.3 [69].

3.2 Head Tracking using Active Shape Models

Active Shape Models (ASM) have been introduced by T. Cootes and colleagues in [47] to devise a method for locating an object inside an image based on its projected 2D shape. The ASM were presented as an improvement of the Active Contour Models [52] (also known as snakes), which are formulated as an energy minimisation problem. Snakes fails at capturing the peculiar modes of variation in a category of shapes. The key idea behind the ASM is that an instant of the searched shape cannot deform much differently from the model learned from a training set of properly annotated examples. The training dataset is a collection of face pictures and related landmark coordinates depicting local salient features (in Figure 27 two examples of labelled faces).

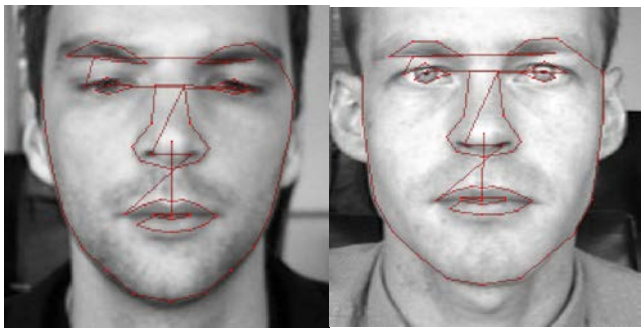


Figure 27. Labelled points on the training set.

All the shapes in the training set are first aligned to each other by minimising the average distance between each other hence giving shape the so-called point distribution cloud at each landmark position. The covariance matrix S computed from all aligned points catches such behaviour and all the pairwise correlations between landmarks. The Principal Component Analysis (PCA) decomposition of S allows finding the principal modes that govern the landmarks displacements. In brief the PCA applied to a covariance matrix computed on a set of observations in R^N finds the subspace of R^N explaining most of the variability in the data. The PCA implicitly assumes a Gaussian as the source of observations. More details on the method have been reported in [66]

The original C++ library implementing the facetracking, the STASM library [55], has been replaced by the Face Tracker library by Jason Saragih [54] that is at present the state of the art of non-commercial facetrackers available.

The Face Tracker library is integrated in the facetrackerCLM plugin described in the accompanying deliverable D4.6b.

3.3 Eye State Detection

The Eye State detection library is a new entry in the SVM. It has been introduced to cope with those situations in which the interaction through head movements is not feasible. In these cases, detecting eye blinks is still a viable option to implement the desired interaction scheme. Blinking per se is not a usable signal since it belongs to a natural and spontaneous behaviour. Rather we are interested in explicit (and prolonged) eyes closures since they can be exploited to detect voluntary signals and used to trigger an event. The implemented method relies solely on images from a standard USB camera capturing the whole face. In

order to discriminate the state of the eyes we start from the natural hypothesis that the appearance of each eye can be in only two states: Opened or Closed. This allows us to interpret the problem as a supervised binary classification and use machine-learning tools to learn how to associate each sample to its most probable class. In short we choose to use the whole appearance of the eye to train a SVM classifier (in this case SVM refers to Support Vector Machine) trained on a low dimensional feature space obtained by the PCA analysis of the whole samples set. The whole process will be presented in three steps: *samples acquisition*, *feature extraction* and *training* of the classifier.

3.3.1 Recording the samples

The training of a supervised classifier requires the availability of a set of labelled samples. Therefore in our case the goal is to have a set of eyes appearances homogeneous with respect to scale, orientation, luminance level and resolution each associated to their respective true state (Open/Close). Therefore the first objective after locating the face is to determine the rectangles that enclose the eyes and apply to each of them a transformation compensating for scale and orientation.

We take advantage of the deformable face model introduced in 3.2 because it allows us to identify directly the outer and inner corners of the eyes. The deformable model is in fact a list of image points each representing a facial feature. In the following we illustrate the implemented procedure to extract one of the eyes. The same steps are then repeated for the other one.

The orientation normalisation is performed through an affine rotation of the image plane around the midpoint between the eye corners compensating for the estimated rotation of the eye. The affine (2D) rotation matrix is parameterised by an angle. The coordinates of the corners of the eye define this angle.

Figure 28. On the left the original image. On the right the corresponding image compensated in rotation. Now the eyes lie approximately on a horizontal line.

Once performed, the eye will appear as horizontal (while the rest of the image will appear as tilted by the opposite angle). Now it remains to crop the rectangle of interest that encloses the eye and rescale the resulting patch to a predefined size. In order to get the same resolution regardless of the apparent dimension of the eye this operation is parameterized by the desired ratio between height and width of the patch: **ρ.** The width and height of the ROI

are thus determined based on the current distance between the eye corners and finally the patch is resized to the target resolution. Figure 29 shows an example of the patches extracted from the image even when the head is tilted at a significant angle. Once the two patches have been resized, one of the two gets mirrored w.r.t. the vertical axe (left-right) in order to align the two patches in a coherent frame and avoiding the need to train a distinct classifier for the left and right eye. In this way we get the additional advantage to be able to extract two eyes samples from a single image. Of course in the recognition phase we need to perform the same operations in order to represent the test inputs in the same way.

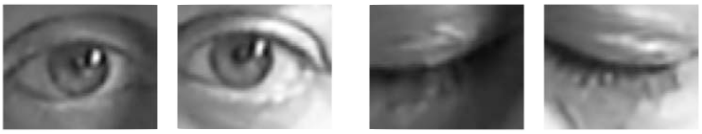


Figure 29. ROIs corresponding to Open eyes and Closed eyes.

3.3.2 Feature Extraction

The feature extraction stage tries to cope with two main problems: illumination variability and high dimensionality of the feature space.

The Retinex [48] [50] is the name for a family of filters designed to provide illumination and colour constancy. Here we use the weighted retinex implementation; each pixel gets a new value as the weighted ratio of the pixel value and a weighted function of its surround [49]. The general shape of the retinex transformation is the following:

$$I_{new}(x,y) = \log(I(x,y)) - \beta(x,y) \cdot \log(F(I(x,y))$$

The ratio between the centre and the surround is implemented as difference between logarithms. weights the mask preventing dark and bright areas in the original image to become greyish. The mask is the surround function that takes into account the presence of edges in the surround. It is usually a weighting function of the local contrast (for example the magnitude of the directional gradients).

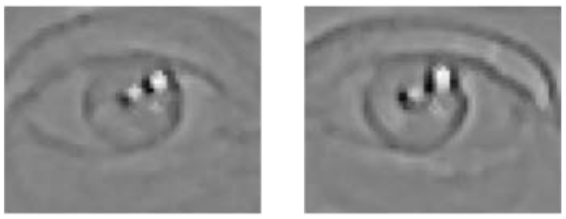


Figure 30. Retinex transformed eye pair.

The PCA is a statistical method that seeks a linear transformation of the original features space that seeks the maximum-variance directions of the distribution of the sample points. When considering only a subset of these directions, in particular those which are responsible for a significative ratio of the total variance, the result is a lower dimensional linear orthogonal subspace representing the projection of the original samples which minimises the reconstruction error in a least square sense. In other words the PCA helps finding a linear projection of the type:

$$y = W^T x, \quad x \in \Re^{nx1}, \quad W \in \Re^{nxt}, \quad y \in \Re^{fx1}$$

The PCA is carried out considering only information up to the 2nd order: the covariance matrix. Figure 31 represents the principal components found by retaining 80% of the total variance. The original space had a dimension of 2880 (48x60 pixel patches).

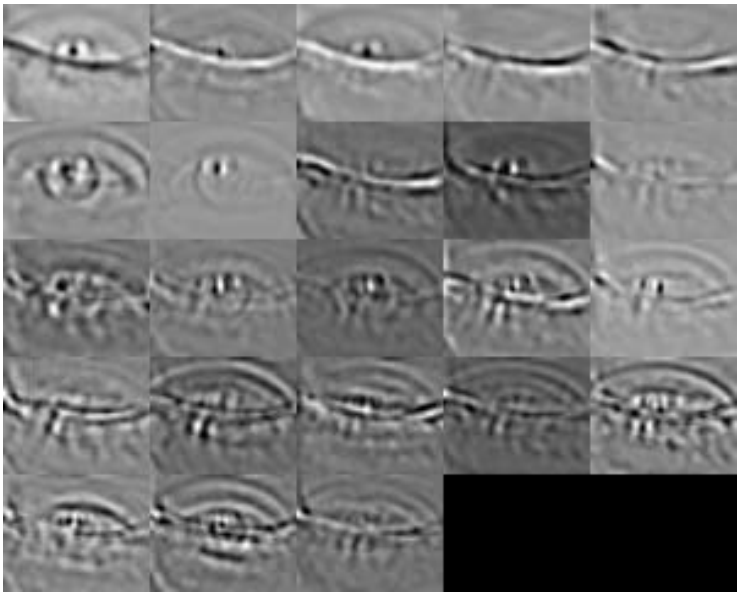


Figure 31. The first 22 Principal Components of a sample set explaining 80% of the total variance. The first patch at the top left corner is the mean of all samples. The original features space had a dimension of 2880 while the projected subspace has 22 dimensions.

3.3.3 SVM training

Support Vector Machines were introduced by Vapnik and Chervonenkis in [27]. The idea is to find the optimal hyperplane to separate a dataset, so that the distance to the nearest datapoint of both classes is maximized. The points spanning the hyperplane are the Support Vectors, hence the name Support Vector Machines. The non-linear version searches the separation manifold in the form:

$$f(x) = sign\left(\sum_i \alpha_i y_i K(x_i, x) + b\right)$$

K(x,y) is a kernel function that transforms data to a high dimensional space, b a bias term, and and are a set of support vectors and their weights. The RBF kernel is a common choice for a very large class of problems.Both linear and non-linear classifiers gave high detection rates on the validation dataset scoring >90% in the case of heterogeneous data (samples acquired from multiple users) and >96% in the case of a training set acquired by a single user. The latter is the typical context in which the detector will work. However the quality of the training data is crucial. Care should be taken when acquiring samples since a non-correct labeling (tagging samples as open when instead they depict closed eyes) will drop quite consistently the recognition rate.

3.4 Wii-Mote Calibration and Wii-Mote Pose Solver

The folder *ARE\components\libraries\upmc\calibrateWiimote* contains also a set of utilities dedicated to the Wiimote sensor developed as external tools. These routines have been developed to estimate the intrinsic parameters of the wii-mote sensor that is integrated in the IR-eyetracker structure. The main purpose of the wii-mote is to help to estimate the spatial pose of the eyetracker (hence of the head) with respect to a pattern that is in a known spatial relationship with a monitor. Details about the eyetracker can be read in D4.6a. The wii-mote calibration aims at estimating the intrinsic parameters of the sensor that are necessary for the algorithms that outputs the pose of the sensor as a function of four IR-leds point sources automatically detected by the Wii-mote. Regardless of its peculiarity the Wii-mote behaves like a perspective camera therefore the calibration steps required are equivalent to the classic procedure when calibrating a RGB camera. Instead of tracking the corners of a checkerboard we will use directly the coordinates of the detected leds spatially arranged as a planar pattern.

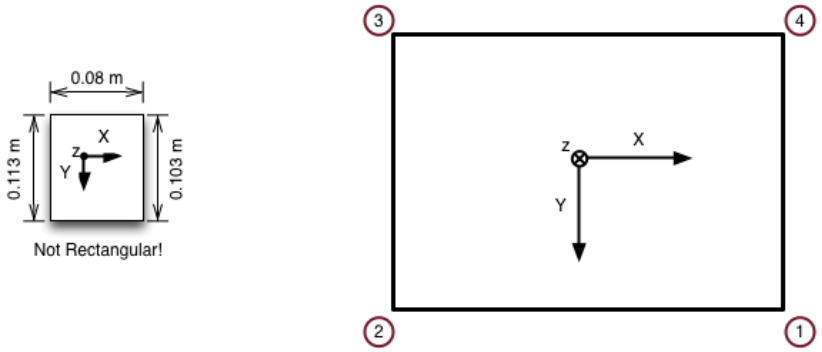


Figure 32 - Example of the calibration pattern used in the calibration phase.

The software should be aware of the geometric spatial configuration oft he pattern. This information for the moment is hardcoded in the source code.

Table 7 shows an excerpt of the sources where the geometry is defined (*main.cpp* from the *ARE\components\libraries\upmc\calibrateWiimote\calibrateWiimote* folder).

```
struct calibObj{
        static cv::Point3f topLeft;
        static cv::Point3f topRight;
        static cv::Point3f bottomLeft;
        static cv::Point3f bottomRight;
};
//Initialize
cv::Point3f calibObj::topRight=cv::Point3f(0.04f, -0.0565f, 0.f);//x4, y4
cv::Point3f calibObj::topLeft=cv::Point3f(-0.04f, -0.0565f, 0.f); //x3, y3
cv::Point3f calibObj::bottomLeft=cv::Point3f(-0.04f, 0.0565f, 0.f);//x2, y2
cv::Point3f calibObj::bottomRight=cv::Point3f(0.04f, 0.0465f, 0.f);//x1, y1
```

Table 7 – The lines in the code that describe the geometric configuration oft he calibration pattern.

The software except to find a file named "calibration.txt" in the same folder as the executable containing the coordinates of the detected points. A negative value means that one or more points haven't been correctly detected. The outcome of the calibration is a YAML file with the intrinsic parameters of the Wii-Mote as a perspective camera. An example in Table 8:

```
%YAML:1.0
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 1.3581483039874283e+003, 0., 4.9112791167817181e+002, 0.,
       1.3879028944794131e+003, 6.1857192824889353e+002, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 5
   cols: 1
   dt: d
   data: [ 1.5767158597068509e-001, -5.9983565185225995e-001,
       5.5242499538168031e-002, -2.3951193086657139e-002,
       1.4712672363690042e+000 ]
imageSize: [ 1024, 768 ]
avg_reprojection_error: 7.0563736054987303e-001
```

Table 8 – The final output of the calibration step.

Other available information are the average squared reprojection error (in pixels) and a video showing the reprojection result through the estimated calibration estimate. See Figure 33 for an example.

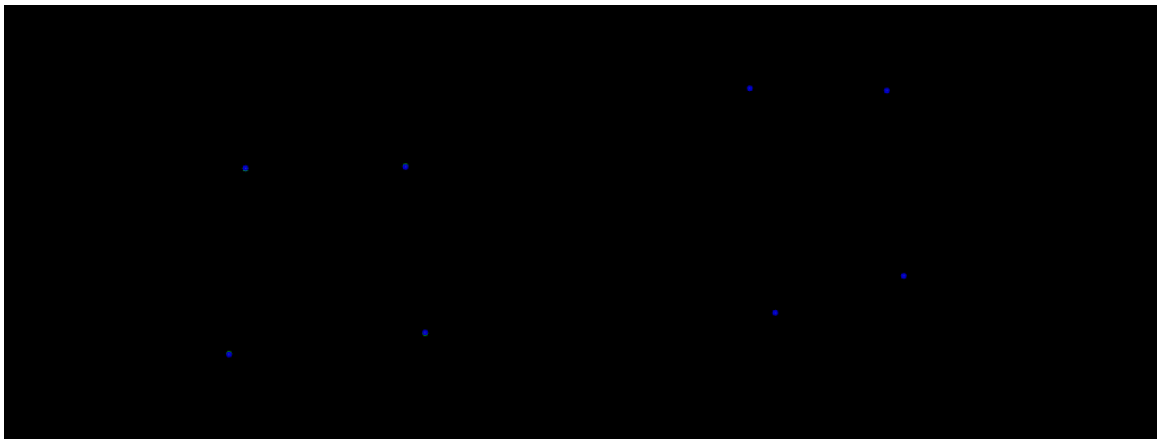


Figure 33 - Reprojection examples: the true detected coordinates and the reprojection match up to a subpixel level (they appear as a single point).

The Wiimote pose solver is an utility class whose use is illustrated in an example project also contained in the main project for VC2010. The pose solver functionality is encapsulated in the class **upmc::wiiPoseSolver.** The solver needs the estimated intrinsic calibration file in ordert o be able to estimate the sensor pose from (at least) 4 known 3D points whose 2D projection coordinates have being detected by the Wiimote.

Table 9 shows a code snippet for declaring and configuring the solver routine. The upmc::wiiPoseSolver::solve method requires 2 input arguments: the known spatial 3D coordinates and their corresponding 2D projections. The resulting pose is given in terms of a rigid transformation R (rotation) and T (translation) of the sensor with respect to the world reference coordinates. Usually it's a common convention to choose the world references frame as to coincide with the reference pattern structure (for example by placing the origin in one corner):

```
upmc::wiiPoseSolver wii;

wii.load(ymlCalib);

[…]

wii.solve(model, imagePoints, R, T);
```

Table 9 – Code snippet for the use of the wiiMoteSolver C++ class.

3.5 Head mounted Eyetracker with Head Pose compensation

For the final prototype of the AsTeRICS SVM, a universal head mounted solution for eye-tracking and head pose estimation has been developed. This system consists of dedicated hardware- and software modules. The hardware components include:

- An adjustable head gear to carry cameras and other sensors which can be comfortably worn for several hours
- 3d-printed parts to allow flexible adjustment of the camera(s), housing of the electronic components and mounting on an external LED-reference frame
- The SensorBoard PCB, which contains IMU sensors with 9 degrees of freedom (3-axis accelerometer, 3-axis gyroscope, 3-axis compass module), a pressure sensor (to allow additional sip-puff control for head/eye-tracking applications) an optical sensor for IR-LEDs and a microcontroller which delivers all sensor information via a USB connection to the host system (usually the host system is the AsTeRICS Personal Platform).

3.5.1 Developed Hardware and Software Components - Overview

Figure 34 shows a block diagram of the hardware and software components for the head mounted SVM. The hardware developments will be described in detail in deliverable D3.4 and are only briefly outlined here to illustrate the workflow of the software components:

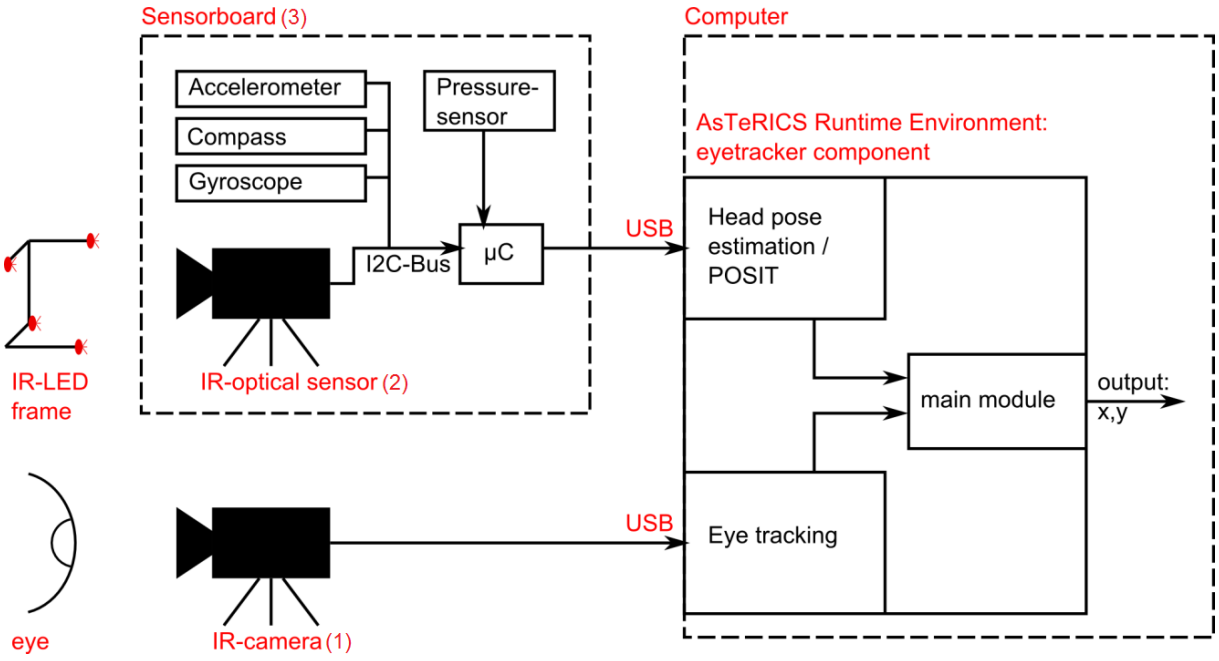


Figure 34. Overview of the electronic system components and software modules for the head mounted eye-tracking and head pose estimation

3.5.2 Flexible Head Mount

The flexible head-mount carries the hardware components so that they are easily wearable. It is a result of several re-design phases using rapid prototyping tools and 3d-printers (see D3.4). The head mount allows attachment of different cameras and sensor modules. Feasible configurations range from an IMU-only operation (no cameras at all) to a fully-featured 3-camera system with two eye-cameras and one scene camera as outlined in D4.3.

Figure 35 depicts the 3d-model of a configuration which can be used for eye-tracking with head pose correction, utilizing the eye-tracking camera (1), the IR-optical sensor (2) and the SensorBoard (3):

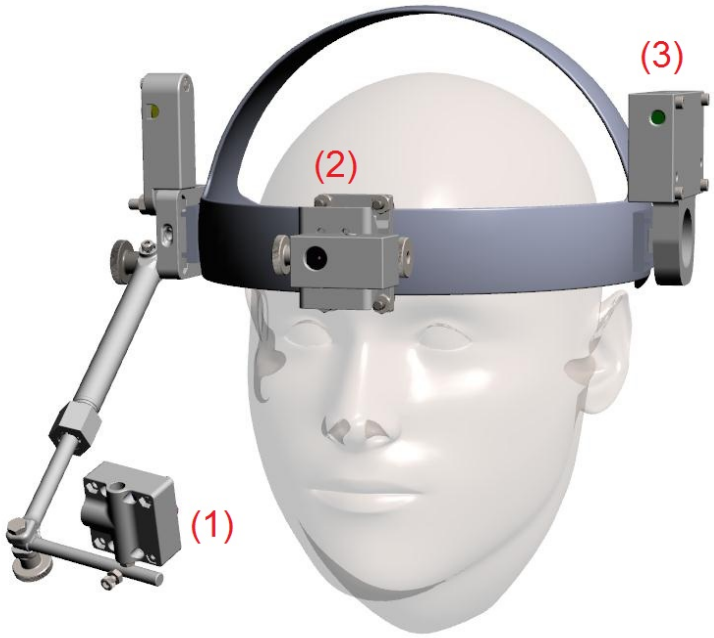


Figure 35. SVM head mounted system: flexible head mount

3.5.3 Possible Use Cases of the Head-Mounted SVM

The developed system is very flexible and scalable and can be used for several applications in the sector of Assistive Technologies, ranging from setups suitable for end-users to pure research configurations. The following list describes several of these applications, starting with low-complexity setups:

- The head-gear and the SensorBoard are used only for the sip/puff sensor functionality. This setup can e.g. be used for scanning selections or sip/puff based mouse- or clicking control. This setup can be realized with the final AsTeRICS prototype.
- The IMU of the SensorBoard is used for the analysis of head movements, to allow head-controlled mouse emulation, optionally with dwell clicks or sip/puff click control. This setup can be realized with the final AsTeRICS prototype.
- The additional use of the eye-tracking camera allows blink detection or selection by eye-movements not related to the computer screen. This setup can be realized with the final AsTeRICS prototype.

- The eye-tracker is calibrated with the computer screen. This setup allows gaze estimation and cursor positioning to the estimated gaze point, if no head movement occurs. This setup can be realized with the final AsTeRICS prototype.

- Additionally to the calibrated eye-tracker, the external LED-frame and the optical IR-sensors are used for head pose estimation. Thus, head movements in a certain range can be compensated to improve the gaze-estimation quality for cursor control. Partially realized in the final AsTeRICS prototype.

- A fully featured setup with 3 cameras is used to perform object localisation in 3d-space (experimental, not completely implemented in the final AsTeRICS prototype).

As can be seen above, some applications can be set up for end-users very easily and work without any calibration. Other applications require different calibration procedures for the eye-tracking.

The fully featured setup with three cameras can be considered as an experimental research configuration which is currently not feasible for end-users. A suitable application has to be chosen according to the user requirements, individual capabilities and the technical knowledge of the care personal, to avoid frustration of the primary or secondary users.

3.5.4 Software / Firmware for the SensorBoard

The SensorBoard is a hardware module which has been developed at FHTW during the WP3 efforts for the final AsTeRICS prototype, as described in D4.3. The firmware programmed into the microcontroller of the SensorBoard provides a virtual serial port implementation for the USB interface. Thus, the SensorBoard serves as a Communication Interface Module (CIM) and uses the CIM-protocol for communication with the AsTeRICS Runtime Environment (ARE).

Data from the SensorBoard can be separately accessed via the dedicated AsTeRICS component as shown in Figure 36. The accelerometer, gyroscope and compass each have separate outputs for the x, y and z axis with the following format:

- Accelerometer: 10-Bit with a range of ± 2 g
- Gyroscope: 16-Bit and range of ± 2000 degrees per second
- Compass: 12-Bit and recommended range of ± 1.3 Ga
- The output of the pressure sensor is 10-Bit with a range of ± 7 kPa.
- pt1X, pt1Y, pt2X … pt3Y are three pairs of the blob detection output from the IR-optical sensor. The fourth pair is internally processed by the component without a visible output port. The values of the port outputs are in relation to the detected blob locations. A blob in the right upper corner from the sensor's view will have the coordinate 0, 0. A blob in the lower left corner will have the coordinate 1024, 768. The initial values of the output ports are 65535 as long as no blob is detected.

Developed AsTeRICS Plugins

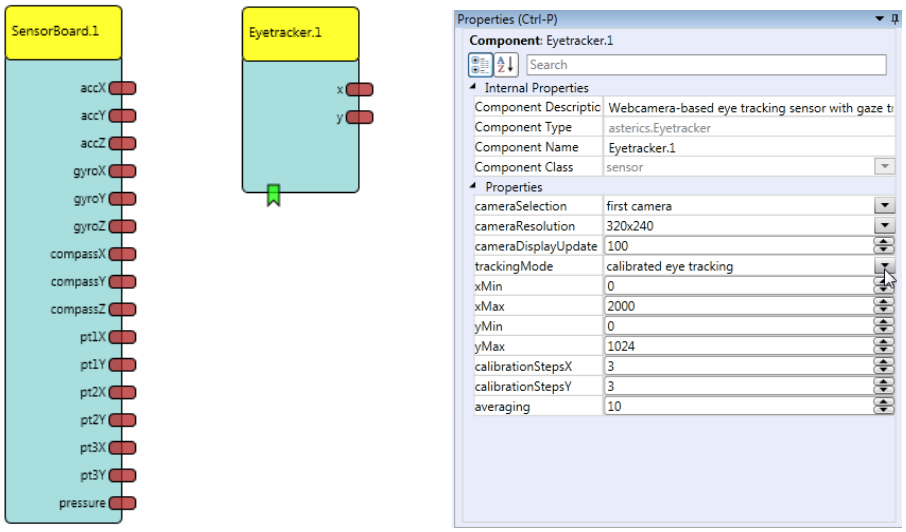


Figure 36. SensorBoard and EyeTracker components (left), properties of the EyeTracker component (right)

For the AsTeRICS Runtime Environment, two components have been developed which utilize the SensorBoard, as shown in Figure 36. The SensorBoard component has output ports for all sensor values, including IMU data, IR-tracking points of the optical IR sensor and the pressure sensor value.

The EyeTracker component fetches the data from the SensorBoard and combines it internally with the pupil-tracking data of the eye camera. The output of the EyeTracker plugin are the x- and y-coordinates of the pupil or of the gaze point on the computer screen, depending on the selected mode of operation.

The properties of the EyeTracker plugin include the selection of the camera, the camera resolution, the update rate of the live display, the area of the calibration surface on the screen, the number of calibration points and averaging samples, and the tracking mode (“only blob tracking”, “calibrated eye tracking”, “calibrated eye tracking with head pose estimation”). Two Event Listeners can be used to start the calibrating procedure or to display the settings of the selected camera. In the live display window of the camera shows the region of interest (ROI).

3.5.5 Algorithm for Pupil-Tracking

For the pupil-tracking, the eye images are recorded with an infrared-camera which is basically a modified webcam (the infrared filter has been replaced by a visible light filter and the original lens has been replaced by a macro lens with standard M12 mount). For illumination of the eye, two wide angle IR-LEDs have been attached to the camera. The light intensity can be controlled via a potentiometer.

The captured frames are sent to the host computer via USB. The EyeTracker plugin analyses these frames in a native C++ thread to detect the pupil’s position and sends the results back to the JAVA main module for further processing.

The IR-camera is accessed by using the VideoInput library by Theodore Watson. With this library, multiple connected cameras can be detected and chosen for image acquisition. Camera settings i.e. brightness, contrast, saturation and gamma can be changed via an API, or a graphical camera settings window can be launched during runtime.

A separate thread is used to display and analyse each captured frame from the camera. On program start-up, a window is opened to display the captured frames from the IR-camera. In the display window, a green rectangle represents the region of interest (ROI). This ROI shows the area of the frames which are to be further analysed and thus reducing the processor load by limiting the amount of pixels to be analysed. The ROI can be manually set by pressing and holding “Ctrl” while using the left mouse button to draw the rectangle.

Each frame is then converted into a greyscale image and the ROI is analysed with the Maximally Stable Extremal Region (MSER) method. A simplified explanation how the MSER method works is as follows: when applying a binary threshold to the image, a pixel would be either black or white. All black pixels are then grouped with their adjacent black pixels so that Extremal Regions are formed for a defined threshold level. The size and form of such Extremal Regions vary for different threshold levels. The MSER method uses the different threshold levels and searches for the Extremal Regions which stay maximally stable for multiple threshold levels. Thus the Maximally Stable Extremal Regions can be found.

Example: One MSER in Figure 37. Could be i.e. the pupil region as it has only slight changes between the threshold levels 5 to 25.

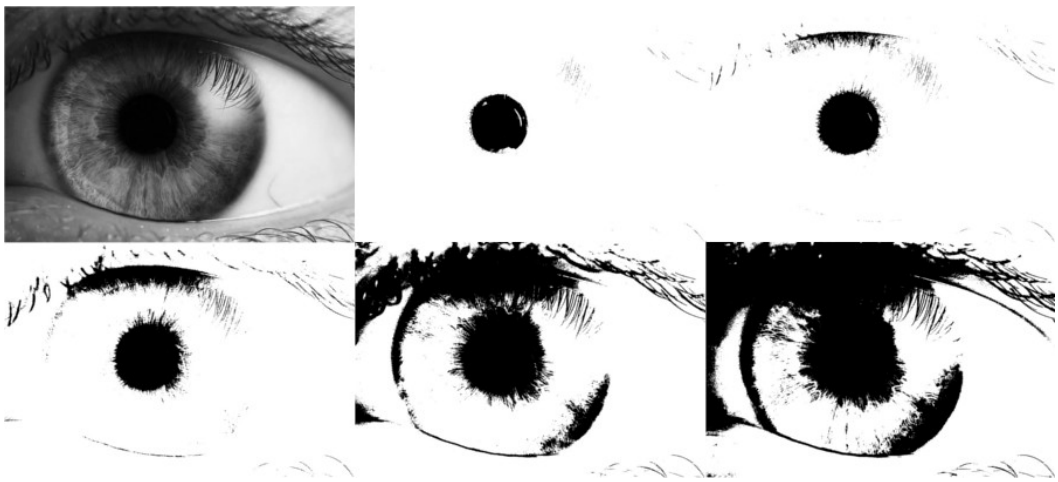


Figure 37. top left: greyscale image of an eye; threshold variants of the greyscale image with following (8-bit) threshold levels: 5, 15, 25, 50, 70 (from top middle to bottom right) [original image: Petr Novák, Wikipedia]

As each frame could have several MSERs, the correct one which matches the pupil best has to be found. For this purpose, the roundness of each MSER is checked and the region with the maximum roundness is assumed to be the pupil. The coordinates of the detected pupil are then sent to the main module by using the Java Native Interface (JNI).

To implement the pupil detection, the following libraries were used:

- CImg: to display the captured frames on the computer screen and to handle the mouse and keyboard input (for the ROI selection)

- OpenCV: for drawing functions (rectangles, ellipses), conversion to greyscale, MSER method and the roundness check
- videoInput: for IR-camera control

3.5.6 Calibration of On-Screen Gaze Estimation based upon Pupil locations

In general, the gaze point is defined by the head position, the head orientation and the eyeball orientation. Thus, various head-poses and eye orientations (gaze directions) could result in the same gaze point [46]:

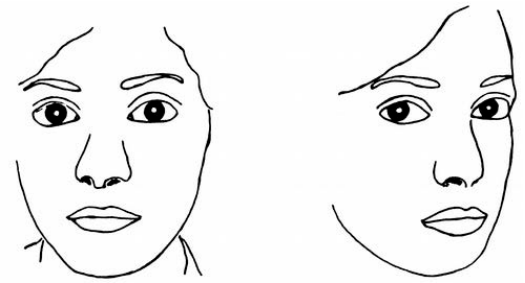


Figure 38. Head pose and eye orientation (gaze direction) define the gaze point

In course of the accomplished work, a linear mapping of eyeball locations to screen coordinates with error compensation due to head pose estimations was implemented. In upcoming future work, the calculation of the gaze direction involving head pose and eyeball locations via a 3d ray-tracing based approach will be investigated.

Linear mapping of eye- to screen coordinates

First, the x- and y- coordinates of the pupil, which have been acquired by using the MSER algorithm (see section 3.5.5), and the corresponding gaze locations on the screen are collected. For this purpose, the user has to look at calibration points on the screen and the respective pupil coordinates are stored in a matrix. The number of the calibration points can be adjusted – more calibration points yield a better online processing result, but the calibration procedure takes longer.

Please note that the calibration has to be performed every time a user sets up the head-mounted camera and starts the eye tracking, as even the slightest change in the relative positioning of the eye-camera and the eye leads to considerable differences in the tracking results. Thus, a compromise between good calibration results and a short calibration time is often desired. Nine calibration points were frequently used in the setups, where a matrix of three horizontal and three vertical gaze points is acquired.

The following figure shows the calibration points and the associated pupil tracking results:

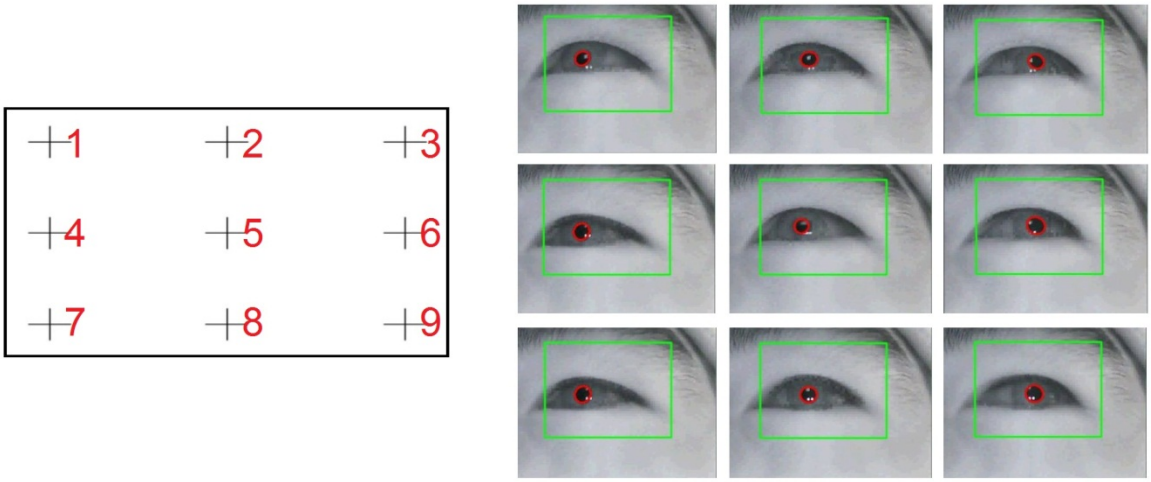


Figure 39. Tracked eyeball/pupil positions for nine calibration points – from top left to bottom right

The aim of the calibration procedure is to provide data for the online operation of the gaze-tracker, so that the current pupil location can be mapped to screen coordinates, based on the calibration points. To achieve this, a linear approximation algorithm is used. Linear approximation will not give optimal results because the eyeball is not a planar surface and the optical system of the camera introduces additional distortions of proportions. On the other hand, a mathematical modelling of the 3d-surface of the eyeball would lead to other complications, from the correct alignment of the pivot point of the eye-camera to individual differences of the eyeball anatomy. As the quality of the linear approximation results depends only on the number of calibration points, it is possible to obtain a reasonable accuracy by utilization of a sufficient number of calibration points for a desired application.

The following assumptions have been taken for the linear approximation:

- The computer screen represents a Cartesian coordinate system, where the x and y coordinates are limited by the screen resolution
- The eyeball / pupil movement has a non-linear relation to the screen gaze points, where distances on the gaze surface (the computer screen) are not equally proportional to distances of the measured pupil locations, depending on the current pupil location.
- The linear approximation of a current gaze point on the computer screen uses the data of the recorded neighbouring calibration gaze points
- The measured x – coordinate of the pupil increases if the x-coordinate of the on-screen calibration point increases (=moves to the right), and vice versa
- The measured y – coordinate of the pupil increases if the y-coordinate of the on-screen calibration point increases (=moves down), and vice versa
- A minimum of 4 calibration points are measured (located at the screen corners)
- The calibration points are equally distributed over the computer screen

Using these assumptions, the following linear approximation model for a tracked pupil location and its neighbour calibration points can be created:

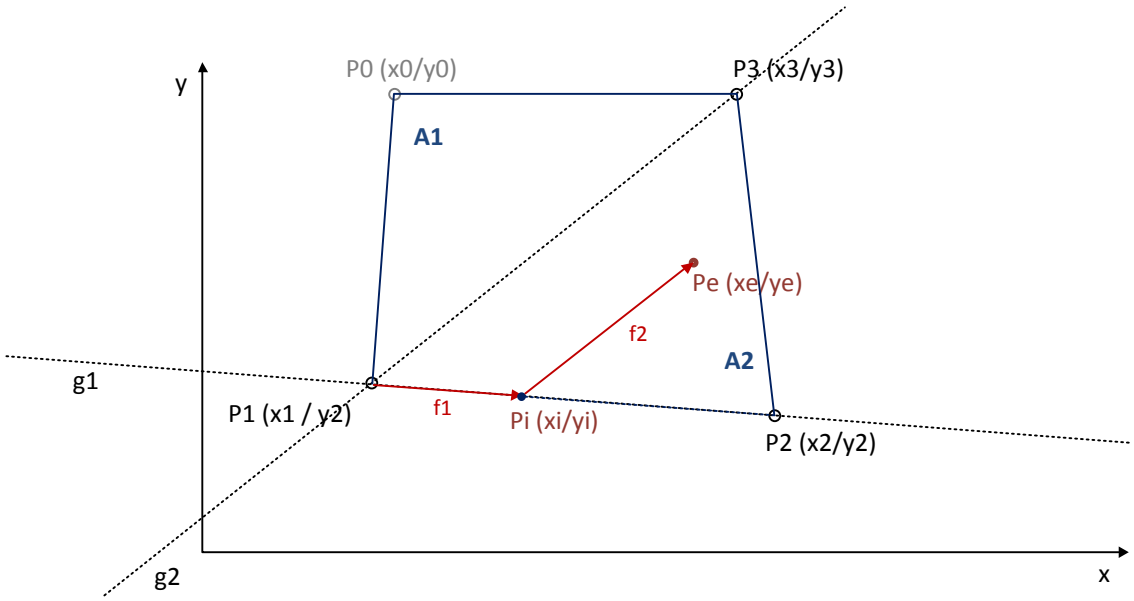


Figure 40. Linear approximation model for pupil location (Pe) and surrounding calibration points

In the above figure, Pe (xe/ye) represents the currently measured pupil position, where P0 – P3 are the nearest pupil positions which have been recorded for the calibration points. These neighbour points can be found by a simple search loop with computation effort O(N).

For the neighbour points, the screen coordinates ($P0_{Screen}$ – $P3_{Screen}$) are known due to the calibration procedure. The approximation uses three neighbour points, depending on the location of Pe: If Pe is located in the area of A1, (P0, P1 and P3) will be used. If Pe is located in A2, (P1, P2 and P3) will be used.

The corresponding screen coordinates of the gaze point (Pe_{Screen}) can be calculated from the calibration points as follows (for area A2, as shown in Figure 40):

$$g1: \qquad y1 = k1 * x + d1 \qquad d1 = y1 - k1 * x1 \qquad k1 = \frac{y2 - y1}{x2 - x1}$$

The coordinates of Pi can be found via the intersection of g1 and g2:

$$Pi: \qquad g1 = g2: \qquad k1 * x + d1 \; = \; k2 * x + d2 \qquad xi = \frac{d2 - d1}{k1 - k2} \qquad yi = k2 * xi + d2$$

When the vector f2 (from Pi to the pupil location Pe) is assumed to be parallel to g2, it can be stated that:

$$ye = k2 * x + d2 \qquad d2 = ye - k2 * xe \qquad k2 = \frac{y3 - y1}{x3 - x1}$$

When a planar, cartesian system is assumed for the screen coordinates of the calibration points, the ratios of $f1_x$ to (x2-x1) and the ratio of $f2_x$ to (x3-x1) can be used for the linear approximation of Pe_{Screen}. These ratios are given as follows:

$$rf1 = \frac{xi \; - x1}{x2 - x1} \qquad rf2 = \frac{xe \; - xi}{x3 - x1}$$

Finally the X/Y coordinates of Pe_{Screen} can be approximated:

$$PEscreenX = X1Screen + rf1 * Xstep + f2 * Xstep$$

$$PEscreenY = Y1Screen - rf2 * Ystep$$

where X_{step} and Y_{step} are the distances of the on-screen calibration point in pixels.

The result is of course prone to head movements. If the measured pupil coordinates of Pe do not have four neighbours (because the gaze point is located outside the calibration area), the approximation boils down to a straight line (between two neighbours) or to a corner point of the calibration area

3.5.7 Algorithm for Head Pose Compensation

To match a specific eye position to a gaze point on the computer screen, the head's position must be considered to avoid huge errors by head movements. With the hardware resources of the head-mounted SVM, this can be accomplished by using the IR-optical sensor and an external reference frame with 4 IR-LEDs. The IR-optical sensor captures the locations of the IR-LEDs, and the SensorBoard transfers these locations to the host computer where the head pose estimation module runs. This module uses the OpenCV implementation of the POSIT algorithm (Pose from Orthography and Scaling with ITerations) [71].

The POSIT method requires:

- Four 3D model points of a reference object (given by the external LED frame), where one point must have the coordinate (0, 0, 0)
- The 2D image points of the object(given by the 2d-coordinates of the blob detection by the IR-optical sensor)

The POSIT method requires that each 2D image point is matched to a specific 3D model point. Thus, the blob coordinates from the IR-optical sensor need to be sorted. The sorting method assumes that there is no considerable rotation around the z-axis, so that the upper left blob detected by the sensor is also the upper left IR-LED of the frame. The sorting occurs each time when all 4 blobs are simultaneously detected by the sensor. There is no need to sort continuously as the order of the sensor output stays the same as long as all 4 blobs / IR-LEDs are detected.

The POSIT method is to be called when all 4 IR-LEDs are detected as blobs and assigned to the corresponding 3D model points. The outputs of the POSIT algorithm are the rotation matrix and translation vector of the IR-optical sensor in relation to the IR-frame. Thus by attaching the IR-optical sensor on a user's head the estimated head pose can be acquired.

The rotation matrix is then transformed into a rotation vector which gives the x, y and z rotations in radians. The unit for the translation vector matches the input unit of the 3D model points (in or case millimetres). The rotation vector and the translation vector are then sent over the JNI to the main module where further processing takes place.

The main module gathers the values from the eye tracker and the head pose estimation module and processes them in such a way that the output matches the point on a computer screen where the user is currently looking.

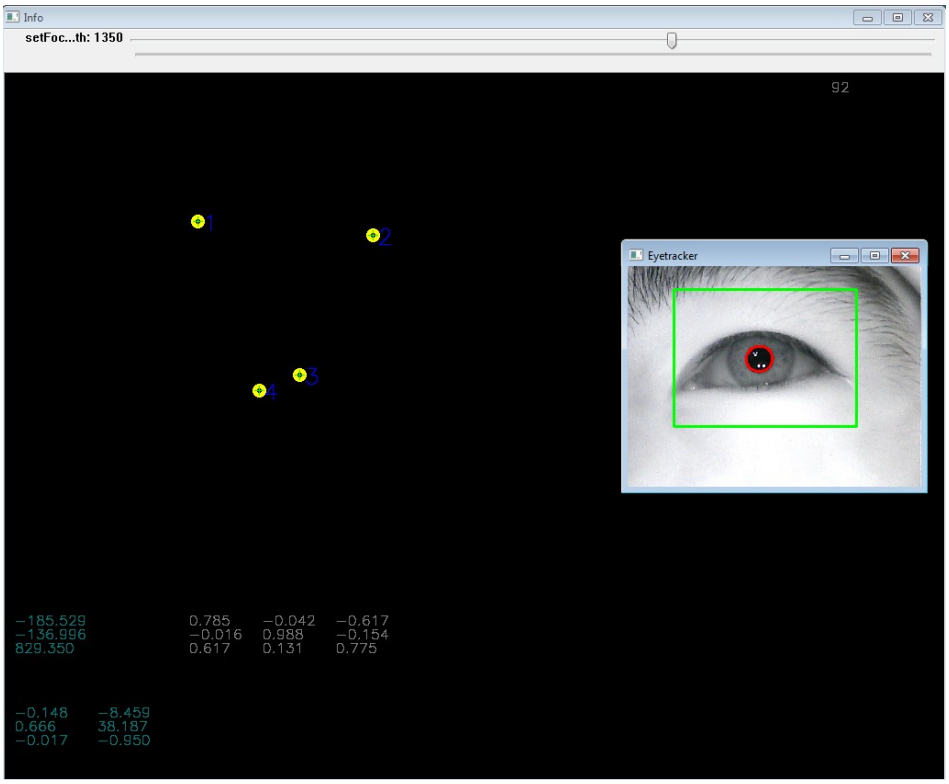


Figure 41. Screenshot of the POSIT algorithm and the integrated eye tracking

Figure 35 shows the combined eye tracking and head pose estimation modules. Both modules calculate independent outputs, where the head pose update frequency depends on the optical-IR sensor / I2C data rate and the POSIT performance (ca. 50 Hz possible on the Personal Platform) and the eye-tracking update rate depends on the frame rate of the infrared camera and the performance of the MSER algorithm (ca. 30 Hz possible on the Personal Platform).

The currently implemented method for head pose compensation in the main module assumes that the user's head does not move during the calibration procedure. The head pose is saved for each successfully captured calibration point and the average head pose during the calibration procedure will be used as the reference head pose.

The difference between the reference head pose and the head pose during eye tracking is used to correct the coordinates of the output, based upon the following formulas:

$$\Delta x = \tan(\Delta rotationY) \cdot \Delta tranxlationZ + \Delta translationX$$

$$\Delta y = \tan(\Delta rotationX) \cdot \Delta translationZ + \Delta translationY$$

The units of Δx and Δy are millimetre and the "delta" values are calculated by subtracting the actual value from the reference value.

The values of Δx and Δy correspond to the estimated "on-screen" gaze deviation caused by head movements. Both Δx and Δy are then converted into the amounts of dots which the mouse pointer should be corrected with. Thereby the coefficient "dots per millimetre" (dpmm) is used. Its value is calculated in the initialisation phase and the formulas for calculation are:

$$diagonalPixels = \sqrt{horizontalResolution^2 + verticalResolution^2}$$

$$dpmm = \frac{diagonalPixels}{screenSize \cdot 25.4}$$

The compensated values of x and y are then sent to the output ports of the EyeTracker component so that other components can use them.

A simple application to control the mouse pointer via eye tracking is shown in Figure 42. The button grid is used to initiate the calibration procedure and to open the camera configuration window. The eye tracking outputs are connected to a component which has the task to set the mouse pointer to the given coordinates.

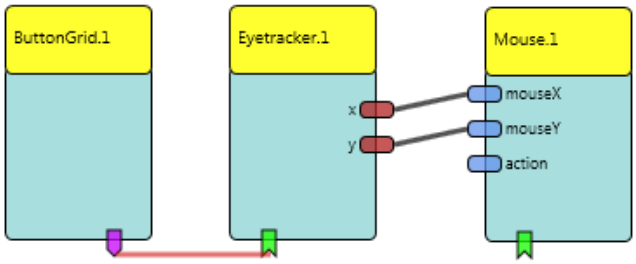


Figure 42. ACS component setup for mouse cursor control via eye tracking

3.5.8 Results and Discussion

The head mounted SVM system has been tested with four male subjects, aged 25 - 40 years, without physical disabilities. Three test sessions per subject have been performed where the calibrated eye-tracking and the calibrated eye-tracking with head pose compensation have been evaluated. The ARE was installed on a standard Laptop (Lenovo Thinkpad W510) with 2GB RAM and SSD drive.

It became clear that the eye-tracking results (in particular the pupil localisation stability using MSER and the maximum roundness of the found regions) performs well only under certain environmental- and lighting conditions. Although infrared illumination and a visible light filter for the eye-camera are used, the setup shows significant differences in the tracking results depending on the amount of direct sunlight in the room, which creates reflections at the cornea (glints) or in eye-glasses worn by the user. Furthermore, the anatomical characteristics of the user influenced the tracking results. However, with some manual adjustments of the camera settings and the camera positioning (which could be easily done due to the flexible 3d-printed boom arm of the head gear) a sufficient tracking stability could be established in 3 of 4 test setups. The greatest problems emerged for the Asian test candidate, where the pupil was partly occluded by the eye-lid in many gaze conditions and could not be reliably detected with the MSER / maximum roundness approach.

The POSIT data integration for head pose compensation proved to be a reasonable addition to the eye-tracking for gaze estimation. Although the head movements and pitch/yaw/roll angles are limited to about 30cm x 30cm and +/-20 degrees respectively, the usability of the gaze estimation was considerably improved compared to the uncompensated approach. The tracking results of the 3 working setups of the head mounted SVM could out-perform the

tracking accuracy of the ITU gaze-tracker [72] and the EyeWriter[8], which are alternative low-cost open source solutions for eye tracking without head pose compensation. A more detailed presentation of results will follow in the WP1 reports.

Considering the stability of the head mount, it can clearly be stated that the flexibility to attach different sensors and the easy adjustment of camera positions is an advantage compared to other existing low-cost solutions - as for example eye-glasses based camera frames. A disadvantage of the head mount is that it cannot be used without modifications for persons lying in bed.

In future works, which can partly be conducted in WP6, alternatives for the simple MSER-based approach for pupil detection will be investigated, to provide glint compensation and better stability in real-life lighting conditions. The second approach for head pose compensation (based upon 3d-ray-tracing) will also be implemented and evaluated.

As a recommendation for the user tests it can be stated that several prerequisites have to be met for a successful application of the head mounted SVM in gaze tracking mode, including the afore mentioned lighting conditions. For users with strong involuntary head movements, the head pose compensation will not work because of the limited field-of-view of the IR-optical sensor. In the user test phase, the results of the head mounted SVM and a commercially available gaze tracker (EyeTech TM-4 mini) will be compared.

[8] http://www.eyewriter.org/

Appendix 1: Appearance-based gaze estimation from a head-mounted system

The work briefly presented in this annex was published by UPMC in [68] and describes the scientific concept and evaluation of a head-mounted gaze tracking approach without infrared illumination. For futher information and references please refer to this document.

Introduction. Eye tracking and gaze estimation models have seen an increasingly interest, in particular for human behavior analysis and human-computer interaction. To estimate the gaze, two configurations are possible: *remote* systems that have to deal with 6 degree-of-freedom head pose estimation and *head-mounted* systems which instead benefit from head-pose invariance and are well-suited for mobile applications.

*Remote system***.** Different appearance-based gaze estimation schemes were proposed using a multilayer Neural Network, an appearance-manifold or a sparse and semi-supervised Gaussian Process Regression (S^3GP) to take account of unlabelled training samples. Recent works also estimate the gaze by avoiding explicit personal calibration and solve the problem using Bayesian inference.

Head-mounted system. While most head-mounted gaze trackers make use of infrared illumination to easily track the pupil, there exist few works on estimating the gaze under natural light. Feature-based methods in natural light rely on tracking the iris and applying a second-order polynomial regression. Recently, a gaze estimation method was proposed and is based on illumination normalization thanks to a modified Retinex model. Hence, it is able to cope with moderate change illumination and can be used outdoors compared to infrared-based gaze tracker.

Methodology. In our work, we present an appearance-based gaze estimation method for a head-mounted eye tracker. The idea is to extract discriminative image descriptors with respect to gaze before applying a regression scheme. We employ multilevel Histogram of Oriented Gradients (HOG) features as our appearance descriptor. To learn the mapping between eye appearance and gaze coordinates, two learning-based approaches are evaluated: Support Vector Regression (SVR) and Relevance Vector Regression (RVR).

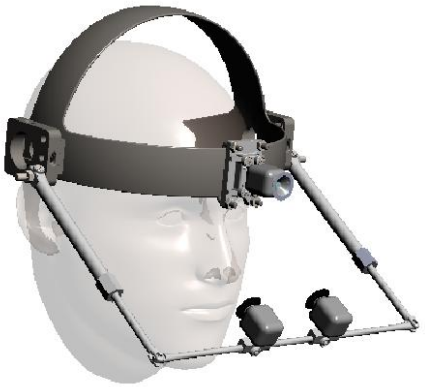


Figure 43: 3D-prototype of our head-mounted system

Eye features. State-of-the-art appearance-based gaze estimation methods mostly rely on raster-scanned (illumination-normalized) intensity images as input data. We instead propose to rely on gradient information which showed to be powerful in feature-based approaches. To do so, we take advantage of histogramming and spatial binning via multilevel HOG. HOG are also known to be robust to illumination changes.

Prior to feature extraction, an eye region must be defined. In the first frame, eye corners are manually labeled for both, left and right, eyes. Then, a Region-Of-Interest (ROI) is set in order to define the eye region that will be applied to the next frames

Once the ROI is defined, eye images are first rotated to be aligned horizontally with respect to the eye corners and rescaled to 120x80. Then, gradient magnitudes and orientations are computed thanks to two 1D filters: [-1 0 1] and $[-1\ 0\ 1]^T$. We use *signed* orientation so that features are constrast sensitive.

HOG features are computed over a grid of 1x2, 3x1, 3x2, 6x4 blocks. Within each block, 2x2 cells are built and a 9 orientation bins histogram is formed. Each block is then normalized according to L2-norm. Finally, the descriptor set of both eyes is concatened to obtain a features vector of length *M=2520*. Multilevel HOG features are efficiently computed thanks to integral histograms. Furthermore, multilevel HOG provide a way to extract a low-dimensional features from the high resolution input image.

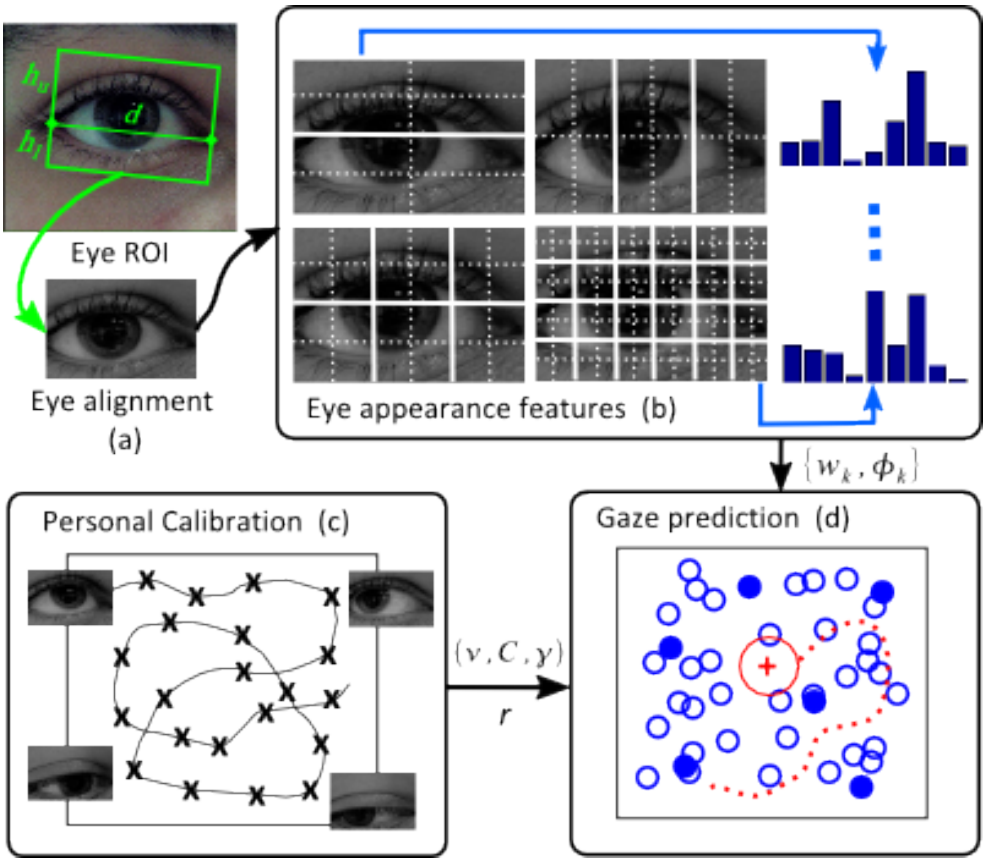


Figure 44: Approach overview for gaze estimation

Setup. To evaluate the proposed method, a low-cost binocular head-mounted eye tracker was built. Two cameras film the eyes and a third camera (with a field of view of 60°) captures images of the scene. Each camera delivers images of size 640x576.

The database consists of 13 subjects and *N* samples per subject are collected. The subject is asked to gaze at a randomly moving target. To simplify offline annotation, the target (a 2x2 chessboard) is automatically detected and in case of detection failure, the target is manually annotated. Two videos per subject were captured. The first video was taken to train the regressors that were tested on the second one. Then, the procedure was repeated by switching the videos to consider different gaze trajectories. The gaze estimation error was computed by taking the average error of the two experiments. In each experiment, the dataset comprises up to 200 training samples and around 400 test samples per subject.

Calibration procedure. For every new subject, a personal calibration is performed in order to collect eye images which implicitly allow to model person-specific eye variations. Regressor parameters are globally optimized to avoid recomputing them for each subject and for a new subject, only the regression model is updated via individual calibration.

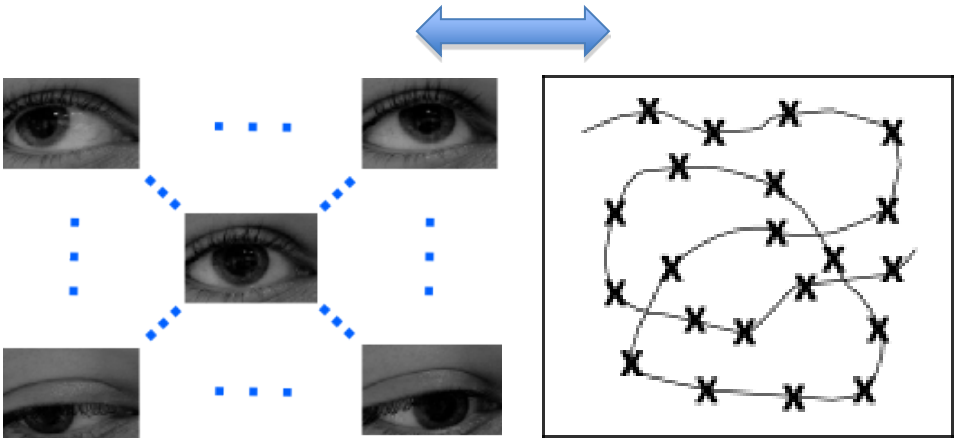


Figure 45. Collecting eye images for calibration

Results and discussion. The performances of both regression models were evaluated in terms of gaze prediction error, robustness to reduced training sets and sparsity of the solution. The Mean Absolute Angular Error (MAAE) and the corresponding standard deviation were computed in visual angle of degrees for each dimension.

Experimental results demonstrate that, despite the high-dimensional input data, our method gives accurate predictions while lowering the number of basis functions and training samples.

Figure presents the results with corresponding error-bars for different number of training samples. Overall, one can see that the error decreases when the number of training samples is large. In terms of the regressor models, SVR and RVR provide comparable results. However, RVR gives a much sparse solution by only keeping a low number (20%-40%) of training samples, while SVR takes most if not all training samples (see Figure 46).

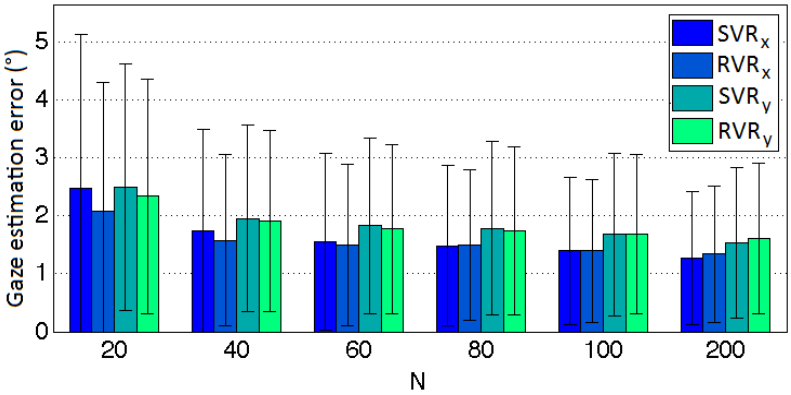


Figure 46. Accuracy vs. number of training samples

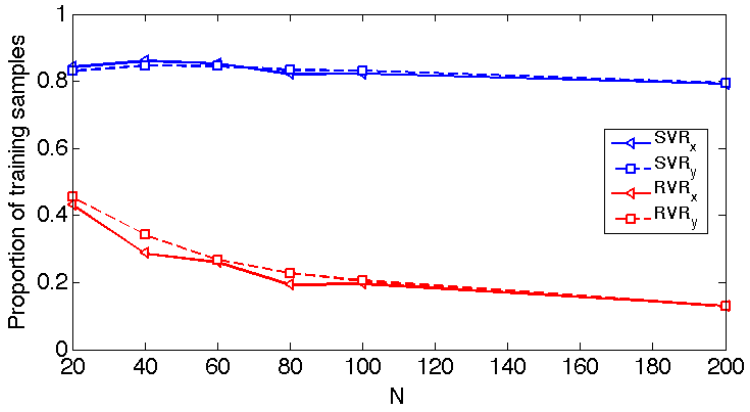


Figure 47. Sparsity results between SVR and RVR

References

1 AsTeRICS Description of Work, technical Annex 1 or the AsTeRICS Grant Agreement

2 TOBI: Tools for Brain Computer Interaction, European Integrated Project; http://www.tobi-project.org/welcome-tobi

3 Biosig: An open source software library for biomedical signal processing, featuring for example the analysis of biosignals; http://biosig.sourceforge.net/

4 BCI2000: A general-purpose system for brain-computer interface (BCI) research; http://www.bci2000.org/BCI2000/Home.html

5 OpenVibe: Software for Brain Computer Interfaces and Real Time Neurosciences; http://openvibe.inria.fr/

6 S. Parker, G. Nussbaum, H. Sonntag, F. Pühretmair et.al. “ENABLE – A view on user's needs” in Proceedings of the 11th International Conference on Computers Helping People with Special Needs (ICCHP), 2008, pp. 1016-1023.

7 AsTeRICS Deliverable D2.3 “Report on API specification for sensors to be integrated to the personal platform”

8 Automation Systems Group, Vienna University of Technology: Calimero NG Library: API for KNXnet/IP remote access in JAVA: http://calimero.sourceforge.net/

9 G. Dornhege, J. D. R. Millan, T. Hinterberger, and D. J. M. eds., Toward Brain-Computer Interfacing (Neural Information Process-ing). The MIT Press, September 2007. [Online]. Available: http://www.worldcat.org/isbn/0262042444

10 A. Kachenoura, L. Albera, L. Senhadji, and P. Comon, ICA: a potential tool for bci systems IEEE Signal Processing Magazine, vol. 25, no. 1, pp. 57–68, 2008.

11 F. Lotte, M. Congedo, A. L´ecuyer, F. Lamarche, and B. Arnaldi, A review of classifica-tion algorithms for eeg-based brain-computer inter-faces. Journal of neural engineering, vol. 4, no. 2, June 2007,http://dx.doi.org/10.1088/1741-2560/4/2/R01

12 Mason, S., Bashashati, A., Fatoure-chi, M., Navarro, K., Birch, and G., A comprehensive survey of brain interface technology designs. Annals of Biomedical Engineering, vol. 35, no. 2, pp. 137–169, February 2007. [Online]. Available: http://dx.doi.org/10.1007/s10439-006-9170-0

13 R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification. New York: John Willey & Sons, Inc., 2001.

14 W. Wei, G. Xiaorong, and G. Shangkai, One-versus-the-rest (OVR) algorithm: An extension of common spatial patterns (CSP) algorithm to multi-class case. Proc. IEEE 27th Annual International Con-ference of the Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005., pp. 2387–2390, 2005.

15 D. McFarland, L. McCane, S. David, and J. Wolpaw, Spatial filter selection for EEG-based communication. Electroencephalography and Clinical Neurophysiology, vol. 103, no. 3, pp. 386–394, September 1997. [Online]. Available: http://dx.doi.org/10.1016/S0013-4694(97)00022-2

16 B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-r. M¨uller, Optimizing spatial filters for robust EEG single-trial analysis. IEEE Signal Proc. Magazine, vol. 25, 2008, pp. 581–607. [Online].Available:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.6768

17 Qin, L. & He, B., A wavelet-based time–frequency analysis approach for classification of motor imagery for brain–computer interface applica-tions, J. Neural Eng., 2005, 2

18 Yong, Y.; Hurley, N. & Silvestre, G., Single-trial EEG classification for brain-computer in-terfaces using wavelet decomposition, European Sig-nal Processing Conference, EUSIPCO 2005, 2005.

19 Qi Xu, Hui Zhou, YongjiWang, Jian Huang, Fuzzy support vector machine for classifica-tion of EEG signals using wavelet-based features. Medical Engineering & Physics 31 (2009) 858–865

20 C. P. Doncaster and A. J. H. Davey, Analysis of Variance and Covariance. How to Choose and Construct Models for the Life Science, 1st ed. Cambridge, August 2007.

21 D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Massachussets: Addison-Wesley Pub. Co., Inc., 1989.

22 G. Pfurtscheller and F. H. L. da Silva, Event-related eeg/meg synchronization and desynchronization: basic principles. Clinical Neu-rophysiology, vol. 110, no. 11, pp. 1842 – 1857, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/B6VNP-3XJSSG7-1/2/919fb705601eee998c08fc2160dc2f99

23 R. Salmelin, M. H¨am¨al¨ainen, M. Kajola, and R. Hari, Functional segregation of movement-related rhythmic activity in the human brain. NeuroImage, vol. 2, no. 4, pp. 237 – 243, 1995.

24 D. McFarland, L. Miner, T. Vaughan, and J. Wolpaw, Mu and beta rhythm topographies during motor imagery and actual movements. Brain Topography, vol. 12, pp. 177–186, 2000, 10.1023/A:1023437823106. [Online]. Available: http://dx.doi.org/10.1023/A:1023437823106

25 G. Pfurtscheller, C. Brunner, A. Schlgl, and F. L. da Silva, “Mu rhythm (de)synchronization and EEG single-trial classification of different motor imagery tasks,” NeuroImage, vol. 31, no. 1, pp. 153 – 159, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6WNP-4J4HK9W-3/2/e2ea7ba6446cd2f634604f1cc42b6080

26 C. S. Herrmann, M. Grigutsch, and N. A. Busch, EEG oscillations and wavelet analy-sis. in Event-related potentials: a methods handbook. T. Handy, Ed. Cambridge, MIT Press, 2005, pp. 229–259.

27 C. Cortes and V. Vapnik, Support-vector networks. Machine Learning, vol. 20, pp. 273–297, 1995, 10.1007/BF00994018. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

28 J. Keller, M. Gray, and J. Givens, A fuzzy k-nearest neighbour algorithm. IEEE Trans. Systems, Man, and Cybernetics, vol. 15, p. 580585, 1985

29 Rakotomamonjy A, Guigue V. BCI competition III: dataset II- ensemble of SVMs for BCI P300 speller. IEEE Trans Biomed Eng. 2008 Mar;55(3):1147-54.

30 J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 20, no. 3, pp. 226-239, 1998.

31 A. Soria-Frisch, "Soft data fusion for computer vision," Ph.D. dissertation, TU Berlin, 2004.

32 M. Grabisch, “Fuzzy measures and integrals for decision making and pattern recognition,” in Fuzzy Structures. Current Trends., Math. Inst., Slovak Academy of Sc, Bratislava, 1997, pp. 7-35.

33 R. R. Yager, “On ordered weighted averaging aggregation operators in multicriteria decision making,” IEEE Trans. on Systems, Man and Cybernetics, 18, pp. 183–190, 1988.

34 L. I. Kuncheva, ""fuzzy" versus "nonfuzzy" in combining classifiers designed by boosting," Fuzzy Systems, IEEE Transactions on, vol. 11, no. 6, pp. 729-741, 2003.

35 M. Sugeno, “The Theory of Fuzzy Integrals and Its Applications,” PhD thesis, Tokyo Institute of Technology, Japan, 1974.

36 Francesco Tortorella (ed.), “Special issue: ROC analysis in pattern recognition,” Pattern Recogn. Lett., vol. 27, no. 8, 2006.

37 J. Cohen, “A coefficient of agreement for nominal scales”. Educational and Psychological Measurement, v. 20, 37-46, 1960.

38 T. Fawcett, “ROC Graphs: Notes and Practical Considerations for Researchers” Machine Learning, 31, 2004.

39 J. Davis and M. Goadrich “The Relationship Between Precision-Recall and ROC Curves”. In Proceedings of the Twenty-Third International Conference on Machine Learning (ICML'06), Pittsburgh, PA., 2006.

40 Parra, L.C.; Christoforou, C.; Gerson, A.D.; Dyrholm, M.; An Luo; Wagner, M.; Philiastides, M.G.; Sajda, P.; , "Spatiotemporal Linear Decoding of Brain State," *Signal Processing Magazine, IEEE* , vol.25, no.1, pp.107-115, 2008.

41 A. Rakotomamonjy and V. Guigue, "BCI competition III: dataset II- ensemble of SVMs for BCI p300 speller." *IEEE transactions on bio-medical engineering*, vol. 55, no. 3, pp. 1147-1154, Mar. 2008.

42 P. Soille, *Morphological Image Analysis: Principles and Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.

43 Margrit Betke, James Gips, Peter Fleming:The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities, IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 10, No. 1, March 2002

44 Christoph Veigl: An Open-Source System for Biosignal- and Camera-Mouse Applications; Submission for the Young Researchers Consortium of the ICCHP 2006, Linz

45 The Intel Open Computer Vision Library: http://www.intel.com/technology/computing/opencv

46 Goldstein, E. Bruce, Journal of Experimental Psychology: Human Perception and Performance, Vol 13(2), May 1987

47 Logitech Webcam Pro 9000 – Product description and features: http://www.logitech.com/de-de/webcam-communications/webcams/devices/5867.

48 E. Land, “The Retinex,” *Amer. Scient.*, vol. 52, no. 2, pp. 247–264, Jun. 1964.

49 E. H. Land, “An alternative technique for the computation of the des- ignator in the Retinex theory of color vision,” *Nat. Acad. Sci. USA*, vol. 83, no. 10, pp. 3078–3080, May 1986.

50 Jobson, DJ, Rahman, Z, Woodell, GA, Properties and performance of a center/surround retinex, IEEE Transactions on image processing, 1997 vol. 6 (3) pp. 451-462

51 Meylan, L, Susstrunk, S, High dynamic range image rendering with a retinex-based adaptive filter, Image Processing, IEEE Transactions on Image Processing , 2006 vol. 15 (9) pp. 2820-2830

52 Cootes et al. Active shape models-their training and application. Computer vision and image understanding (1995) vol. 61 (1) pp. 38-59.

53 Kass et al. Snakes: Active contour models. International Journal of Computer Vision (1988) vol. 1 (4) pp. 321-331

54 Jason M. Saragih, Simon Lucey and Jeffrey F. Cohn, Deformable Model Fitting by Regularized Landmark Mean-Shift, International Journal of Computer Vision, 2011, Volume 91, Number 2, Pages 200-215

55 Milborrow and Nicolls. Locating facial features with an extended active shape model. Computer Vision–ECCV 2008 (2008)

56 Li et al. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. Proceedings (2005) pp. 79-86.

57 Li and Parkhurst. Open-Source Software for Real-Time Visible-Spectrum Eye Tracking. Proceedings (2006) pp. 18-20.

58 Colombo et al. Robust Iris Localization and Tracking based on Constrained Visual Fitting. Proceedings (2007) pp. 454-460.

59 Ryan et al. Match-moving for area-based analysis of eye movements in natural tasks. Proceedings (2010) pp. 235-242.

60 Ryan et al. Limbus/pupil switching for wearable eye tracking under variable lighting conditions. Proceedings (2008) pp. 61-64.

61 Hansen and Pece. Iris Tracking with Feature Free Contours. Proceedings (2003) pp. 208-214.

62 Wu et al. Tracking Iris contour with a 3D eye-model for gaze estimation. Proceedings (2007) pp. 688-697.

63 Loy and Zelinsky. Fast radial symmetry for detecting points of interest. IEEE Trans. on PAMI (2003) vol. 25 (8) pp. 959-973.

64 Isard and Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. IJCV (1998) vol. 29 pp. 5-28.

65 Zhang et al. A Robust Algorithm for Iris Localization Based on Radial Symmetry. Proceedings (2007) pp. 324-327.

66 Okuma et al. A Boosted Particle Filter: Multitarget Detection and Tracking. Proceedings (2004) pp. 28-39.

67 Viola and Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on (2001) vol. 1 pp. 511.

68 Francis Martinez, Andrea Carbone and Edwige Pissaloux.. Radial symmetry guided particle filter for robust iris tracking. To be published in Proc. of Computer Analysis of Images and Patterns (CAIP 2011).

69 AsTeRICS deliverable D4.3 Prototype 1 of Signal Processing Modules, various authors.

70 Ola Friman, I.V., Axel Graser, Multiple Channel Detection of Steady-State Visual Evoked Potentials for Brain-Computer-Interfaces. IEEE Transactions on Biomedical Engineering, 2007. **54**(4).

71 L. S. D. Daniel F. DeMenthon, “Model-Based Object Pose in 25 Lines of Code,” International Journal of Computer Vision, vol. 15, no. 1-2, pp. 123-141, 1995

72 Javier San Agustin, Henrik Skovsgaard, Emilie Mollenbach, Maria Barret, Martin Tall, Dan Witzner Hansen, and John Paulin Hansen. 2010. Evaluation of a low-cost open-source gaze tracker. In Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications (ETRA '10). ACM, New York, NY, USA, 77-80.