Graz University of Technology

Master Thesis

Online optimization of Man-Machine interaction - A recurrently updating Brain-Computer Interface based on Common Spatial Patterns and Random Forest classifier

conducted at the
Institute for Knowledge Discovery
Graz University of Technology, Austria

by
Schwarz Andreas, BSc., 0230572

Supervisor:
Ass Prof. Dipl.-Ing. Dr. Reinhold Scherer

Graz, April 24, 2014

Acknowledgements:

At this place I would like to thank all those people who contributed in their own way of making this thesis possible: First and foremost, I would like to thank Josef Faller and David Steyrl. This thesis is based on their extensive research, and their guidance was imperative for completing this task. Furthermore credit belongs to Reinhold Scherer (Reini), who gave me the chance to work in this field and supervised my research.
None of this would have been possible without Christian Mathis, head of webtourismus.at GmbH, who gave me the opportunity to work for his company at the best conditions one could imagine, so I got the chance to graduate.
To my family, who endured my quite literally interpretation of life-long learning with... serenity, and Thomas Wiesner, who pushed me and kept me motivated.

Finally I want to thank Katharina Haring, for love, patience and motivational whiplashes.

Abstract

Sensorimotor rhythm (SMR) based brain-computer interfaces (BCI) enable the user to control devices by motor imagery, which is the mental rehearsal of hand or feet movement. Ultimately, the BCI should lead to an assistive device for people with severe motor impairments.

Typically, BCI systems require a lengthy period of calibration and user training which can last several months for people with severe motor impairments. This can be exhausting and fatiguing for the user as the training tasks are usually (1) monotonous and (2) without any feedback to encourage the subject. Moreover, frequently those BCI systems lack of sufficient performance to provide decent assistance to the user. Hence, new ways to reduce user training and improve performance have to be found.

This thesis proposes a two class SMR based BCI system, which performs recurrent updates of its parameters during runtime to boost overall performance. Therefore, it is able to provide accurate, positive, visual feedback to the user after just 4 minutes of calibration.

To maximize the discriminability of logarithmic band-power features, common spatial patterns (CSP) along with a filterbank were used. The features were classified by a Random Forest classifier. The system recurrently updated the CSP and classifier models. To reduce the influence of trials contaminated with physiological or non-physiological noise, every training step was preceded by statistical outlier rejection.

In a supporting online study, all 9 novice, able-bodied volunteers performed significantly better than chance with an overall peak accuracy of $84.9 \pm 10.3\%$. This outperforms the performance of state of the art BCI systems, which perform ususally with an peak accuracy of $75 \pm 15\%$. Due to the high performance of the system, it may be an expedient solution in creating an assistive device for people with severe motor impairments.

Kurzfassung

Sensorimotorisch rhythmus-basierte Brain Computer Interfaces (BCI) ermöglichen ihrem Benutzer die Kontrolle von Geräten allein durch die Kraft einer Bewegungsvorstellung. Diese BCI Systeme sollen letztendlich einem bewegungstechnisch stark eingeschränktem Menschen als unterstützendes Hilfsmittel zur Seite stehen, um so mit seiner Umwelt zu interagieren.
Üblicherweise benötigen BCI Systeme ausgedehnte Trainings -und Kalibrierungsperioden, die mehrere Monate andauern können. Die Trainingsparadigmen sind meist recht monoton gestaltet und geben kein Feedback, um die Aufmerksamkeit des Benutzers auf sich zu ziehen. Dementsprechend passiert es häufig, dass die Leistung dieser Systeme unter den Erwartungen bleibt. Daher müssen andere Ansätze gefunden werden um Trainings -und Kalibrierungsperioden zu verkürzen und die Leistung dieser Systeme zu steigern.
Diese Diplomarbeit legt ein 2 Klassen BCI System dar, welches auf Bewegungsvorstellungen beruht. Um die Leistung des Systems zu steigern werden zu vordefinierten Zeitpunkten die Parameter des Systems neu berechnet, was dazu führt, dass das System in der Lage ist, bereits nach etwa 4 Minuten akkurates, positives visuelles Feedback zu geben.
Um die Trennbarkeit der logarithmischen Bandpower Features zu maximieren, wurde ein "Common Spatial Patterns" Filter zusammen mit einer Filterbank implementiert. Die daraus resultierenden Features werden mit einem "Random Forest" klassifiziert. Das System updated das Filtermodel und den Algorithmus in fixen Abständen. Um den Einfluss von physiologischen und nicht physiologischen Störungen in den Bewegungsvorstellungen des Benutzers zu minimieren wurden vor jedem Update die Daten mittels statistischer Methoden untersucht und abnorme Daten ausgeschlossen.
Die Leistung des Systems wurde mittels einer unterstützenden Studie an 9 naïven, gesunden Freiwilligen festgestellt. Alle Personen lagen mit ihrer Leistung deutlich über dem Zufallslevel mit Spitzenwerten von bis zu 84.9 $\pm$ 10.3% (median 78.0%). Diese Leistung liegt deutlich über jener von vergleichbaren modernen BCI Systemen, deren Spitzenwerte bei 75 $\pm$ 15 % liegen. Aufgrund dieser bemerkenswerten Leistung könnte das System einen Lösungsweg für bewegungstechnisch stark eingeschränkte Menschen darstellen.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Contents

1 Introduction

A Brain-Computer-Interface (BCI) is a device which enables the user to interact with its surrounding only by thought (e.g. imagination of a motor task) or attention to a stimuli (e.g. focussing on an external stimulus). In the end, a BCI should ease, or even allow a user with severe motor impairment not only to interact, but to communicate with its environment.
Since the initial steps in this field of science in the early seventies [1], BCI systems have significantly evolved, including not only a narrow banded field of a few neuroscientists, but a vast interdisciplinary community with annual publications in the hundreds [2]. With the growth of the community the variety of different approaches to measure and classify brain activities multiplied.

When recording brain activities, two primary approaches are considered: Invasive measurement, which implies invasive brain surgery for placing electrodes directly on the designated cortex areas (electrocorticography (ECoG)), and non-invasive approaches, where the skin of the user is not penetrated. For recording brain activities non-invasivly, a number of methods are available, where electroencephalography (EEG) is the most common and widely spread in the BCI community.
Neurons in the brain generate and transfer action potentials, called spikes. The spikes itself generate post synaptic potentials (PSP) which are pooled into compound action potentials and can be measured by the EEG on the scalp.
While spikes have amplitudes up to 70 mV, the compound action potentials measured on the scalp are damped by the skullcap and organic matter resulting in a range of 10 to 500 μV. Because of this exceptional weak signal strength, the measured EEG signal is highly vulnerable to any kind of interferences and noise, leading to a very small signal to noise ratio. Dealing with these interferences, also called artefacts, is a major issue for every BCI system.
Nevertheless, several advantages do make the EEG a good candidate for data acquisition in the field of BCI: The EEG is a non-invasive method (1), so no penetration of the skin or extensive surgery is required. All necessary electrodes are placed on the scalp of the user and the system can be operable within minutes. The EEG offers a high temporal resolution (2) - voluntary changes in the brain activity can be measured within milliseconds. The EEG equipment requires comparatively low investment (3) - although the price range for multiple electrode systems is unbounded, standard BCI equipment can be purchased for the price of middle-class car.

The measured brain signals must contain components which can be voluntary modulated by

a user, in order to interact with its environment through a BCI system. Among the investigated phenomena [3], one class deals with different motor imageries (MI), which may be seen as a mental rehearsal of a motor act without executing it. These imageries cause oscillatory changes (power de/increase) in the α (8 - 12Hz) and β bands (13-30 Hz) over defined brain areas [4]. The phenomena is called event-related synchronisation (ERS) or power increase and event-related desynchronisation (ERD) or power decrease, and occurs subjected to the used Motor Imagery, in different locations of the brain. Figure 1.1 shows the sensory homunculus which represents the relative amount of cerebral cortex surface given to every sensory and motor input and output of the human nervous system [5]. ERD/ERS is known since the early seventies, and is explained in detail in [6].

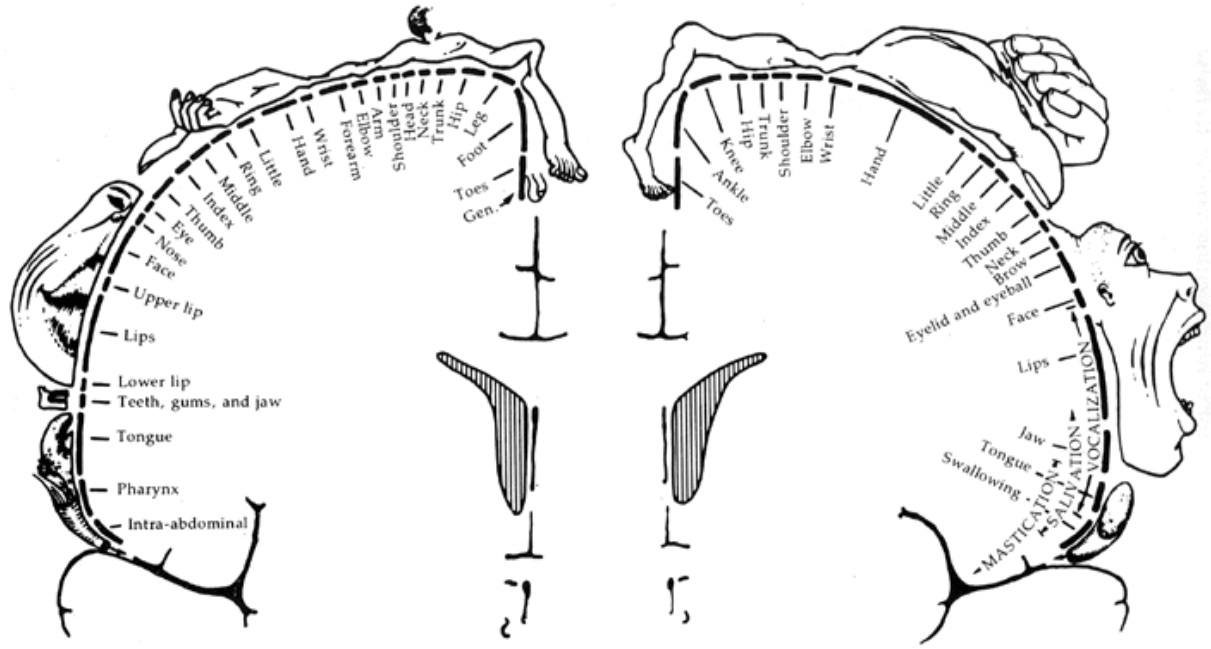


Figure 1.1: Mapping of body parts according to the motor(l.) and somatosensory cortex: Foot and leg are to find medial, hand, face and mouth are to be found lateral [5]

The basis of every BCI system, is shown in figure 1.2. The BCI system acts as a closed loop system which directly engages the subject by giving feedback to the input of the user, therefore a high time resolution (milliseconds) for real-time application is imperative. Signal acquisition may be done - as already described before - invasive or non-invasive.
Preprocessing involves amplifying the signal as well as clean it of all kind of interferences, noise or artefacts. The final stage of the Preprocessing step should deliver a signal which is optimized for extraction of features feasible for describing brain activities.
The main task of the features is to describe the change in the brain activity by the voluntary modulation of the user. The better the features describe the different voluntary modulated brain signals, e.g. different motor imageries, the more promising is the ability to actively control an application using the BCI for the user. For sensory motor rhythms (SMR), like the previous described ERS/ERD, the power information of the signals in the range of mu and beta rhythms seems to be a valid approach. Nevertheless a substantial amount of feature extraction methods are investigated, and can be found in [3] and [7]. The classifier is assigned to discriminate between the voluntary modulated brain activities of the user, using the features extracted from the signal in the previous step. Usually, a large amount of individual training data hast to be recorded for each subject in order to perform in the desired accuracy ranges. Depending on the selected brain activity, the training data may persist over more than one session or in most

cases, has to be redone with every new session. Classification algorithms are common in a vast number of technical branches, so the potential for investigating procedures used in other fields of research is high and encouraging.
The decisions made by the classifier are the input for the application interface, which controls any sort of application allowing the user to interact with its environment using only pre-defined brain activity. The variety in applications is endless, starting from assistive devices for patients with locked-in syndrome or tetraplegics up to e.g. a new input device for the entertainment industry respective computer games [8] [9].

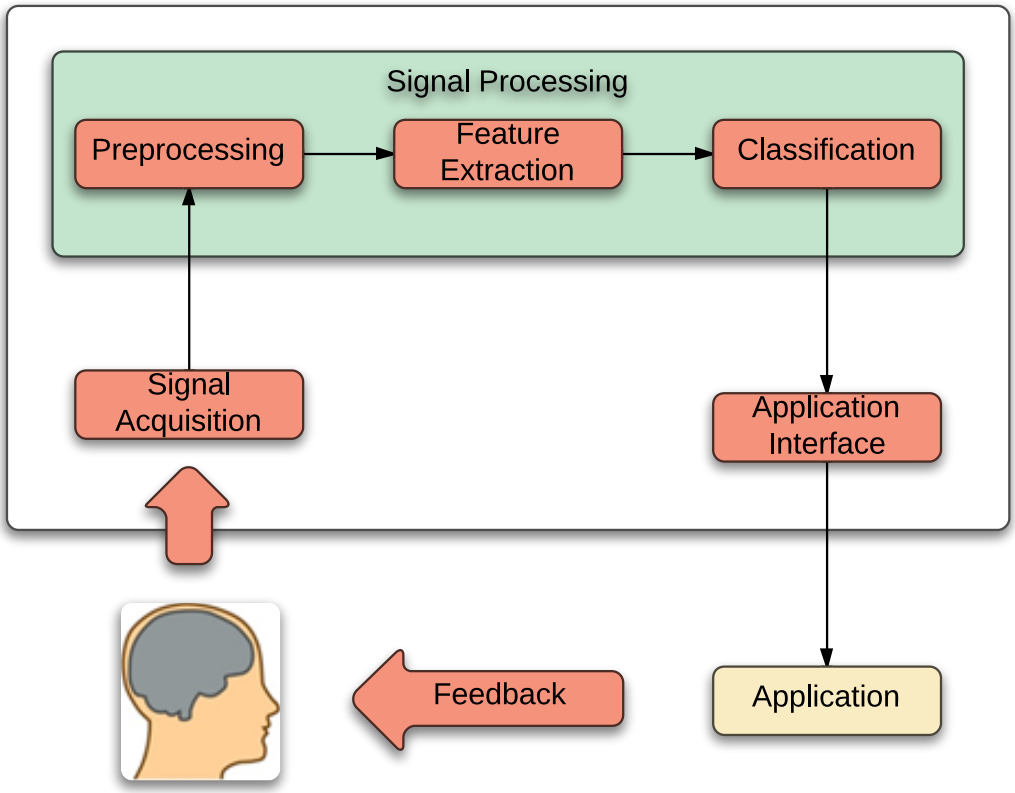


Figure 1.2: Schematic of the general BCI approach

Although state of the art BCI systems use this general template as basis, each has its own innovations to the general system. In 2012 Faller et al. [10] published their approach on an auto calibrating and adapting BCI for MI tasks. The system is based on a standard BCI system using bandpower features [11] (α-band and β-band for three Laplace channels [12]) and linear discriminant analysis (LDA) as classifier.
Conventional BCI systems usually run a lengthy (at least 30 minutes) period of gathering training data for the classifier, without giving any feedback before entering a stage where the subject can interact with the BCI. Training period and the preceding interaction period are in a sequential order. Faller et al extended the standard BCI ("Online System") by an independent block called "Optimizer Instance", where recurrent updates of the classifier using the gathered data

of the "Online System", are calculated and sent back to the Online System.
This enables the system to provide positive reinforced feedback within minutes of training instead of a lengthy training period. Results level in on a peak average of 75 percent over 12 able bodied, naive users.
Steyrl et al. [13], [14] investigated in 2012 the usefulness of a more sophisticated classification method for BCI systems, called Random Forests. He proposes that due to the non-linear behaviour of the EEG patterns, a BCI system would benefit from a more sophisticated, even non-linear classifier model, instead of the commonly used Linear Discriminat Analysis (LDA) classifier. To confirm the hypothesis, a supporting study was done resulting in 76 percent over 12 abled naive users.

Both BCI approaches contain elements which seem innovative and beneficial for further use, but are implemented in a standard BCI system. While Faller et al. use a parallel "Optimizer Instance", their system uses a linear classifier. In reverse, Steyrl et al. use a sophisticated classifier, but relies on lengthy training periods without recurrent updates of an "Optimizer Instance".
The idea is to design a new BCI system which makes use of not only their investigations towards an Optimizer and a non-linear classifier, but to extend it even further by using sophisticated preprocessing methods, namley filterbank common spatial pattern (CSP) [15]. The result would be a new BCI system with a battery of sophisticated methods, which replace all well established and published standard methods.
The aim of this work is to investigate, whether or not a combination of these technologies can be achieved in an expedient way. To evaluate the performance of the approach, a supporting study is considered the most convincing way.
Section 2 starts with a conceptual overview of the the system, and will provide theoretical approaches to the applied technologies, as well as supplemental information of the supporting study. Section 3 will present the findings and results gathered in the supporting study, Section 4 discusses the result in detail and point out issues and features for future investigations. Section 5 concludes the thesis. The appendix holds all critical implementation details and analysing scripts.

2 Methods

2.1 Concept Design

Figure 2.1 displays the concept of the signal processing chain of the proposed system. It should act a as a two class BCI system which uses different motor imageries, motor imagery of the right hand and motor imagery of both feet, as control signals. As input for the system acts a multichannel EEG signal. To boost the discriminability between the classes, a battery of common spatial patterns filter in combination with a bandpass filterbank (FCSP) is used before calculating logarithmic bandpower features. These features are classified by a Random Forest classifier. The classifier delivers a the classlabel, ("right hand" or "both feet") which can be used to calculate feedback for the user. Recurrent updates to the classifier and the FCSP should ensure that the system improves in performance over time.

The idea is to keep the period, where the user does not get feedback, as short as possible. Long training periods with no feedback do wear off the user's attention and motivation. By keeping this period short and delivering feedback to the user as soon as possible, motivation and attention can be kept up. Moreover, based on the feedback the user gets, he should be able actively train the motor imageries, and in turn the system adapt better to the user's input.

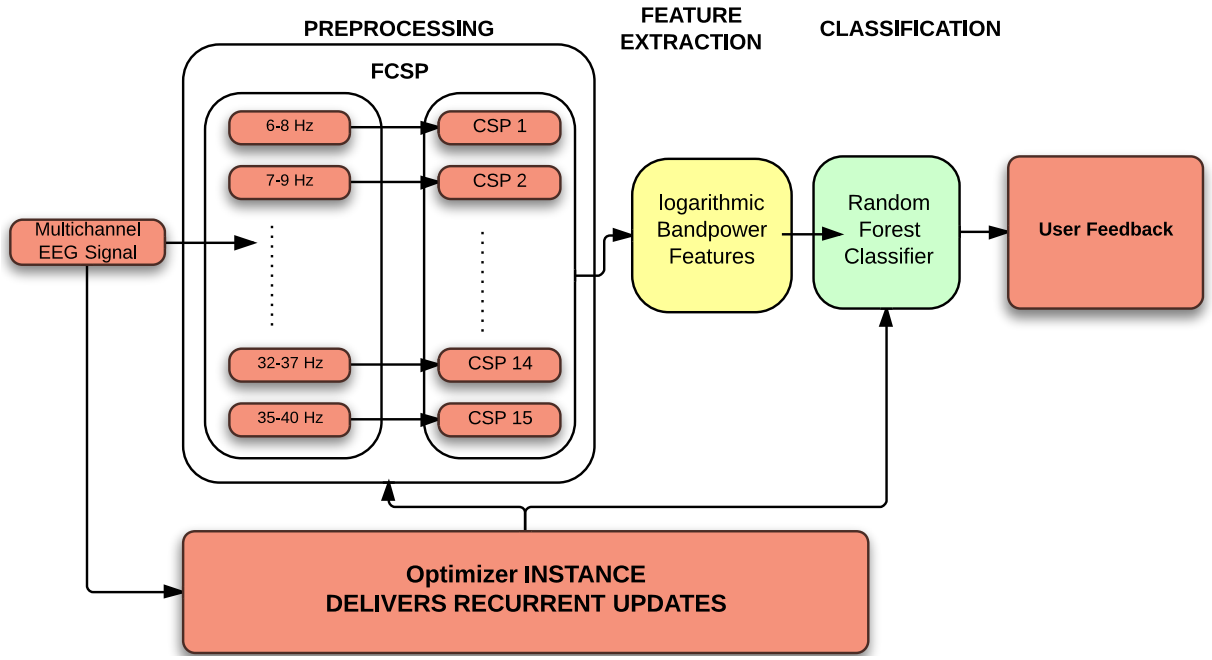


Figure 2.1: Concept overview: Signal chain of the proposed system

2.2 Preprocessing and Feature extraction

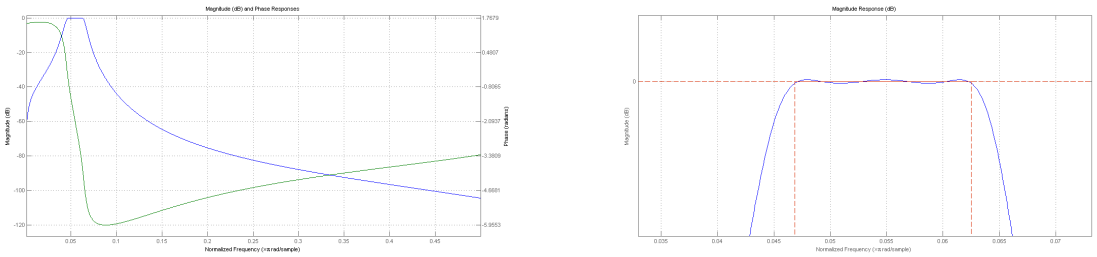
2.2.1 Filters and Filterbank

As already mentioned in the introduction, physiological arguments suggest to investigate μ and β bands. These are especially relevant for extraction of motor imagery activity.
Therefore, the acquired EEG signal is fed into a so called "Filterbank" which essentially is a conglomeration of IIR bandpass filters. The bandpass-filters are designed as IIR filters of order 6, second order filter coefficients have been scaled for reducing chance of overflow. To cover μ and β bands, 15 filters are used which are arranged narrow-banded and in an overlapping way. Table 2.1 shows the individual pass ranges for each filter. In figure 2.2 the filter characteristic, which can be applied to every filter used, is shown.

bandpass	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lower cut-off [Hz]	6	7	8	9	10	11	12	14	17	20	23	26	29	32	35
higher cut-off [Hz]	8	9	10	11	12	13	14	19	22	25	28	31	34	37	40

Table 2.1: cut-off frequencies of the bandpass filterbank

The narrow banded filters allow investigations in relatively small areas of the designated mu and beta bands, the overlapping of the frequencies compensate for the slope of the filter. An additional beneficial factor of this narrow-banded design is that the influence of artefacts, which can and will occur during measurements on human subjects is greatly reduced. Artefacts are induced by muscle activity (EMG), eye-movement (EOG), teeth grinding, sweat or even shivering and can take a great negative influence in the performance of a BCI. The topic of artefacts will be further investigated in the optimizer section.



(a) Magnitude (blue) and Phase (green) response

(b) Magnitude Response, Passband Ripple

Figure 2.2: Stable Magnitude and Phase Response of one of the bandpass filters of the filterbank. 2.2(a) Both Stopband Atten. lie around -44.7dB ; 2.2(b) Passband-Ripple > 0.01 dB, Frequency is normalized between 0 and 1, Sample Rate: 256Hz

2.2.2 Common Spatial Patterns

To boost class separability, the common spatial pattern (CSP) method is applied on each filter output of the filterbank individually. The main motivation is to prepare the filterbank signals to get an optimized class separability when calculating the final features.
CSP is based on a decomposition of the raw EEG signals into spatial patterns, which are extracted from two classes. The classical CSP approach is only defined for two classes. It maximizes the variance of the spatially filtered signals under one condition, while minimizing it for the other condition [16],[17].

Originally developed to discriminate between two different EEGs (normal and abnormal) in the Nineties [18], the method was soon adapted to fit other needs in the field, as well as for optimizing procedures in the field of SMR-BCI [18], [19].
Nowadays there exist research groups who specialise in this method by pushing further investigations, modifications and refinements to CSP as can be seen in [16] (CSSSP, Common Sparse Spectral Spatial Patterns), [20] (SpecCSP, Spectrally Weighted CSP) or [21] (ISSPL, Iterative Spatio-Spectral Patterns Learning). The CSP algorithm has become most popular in the BCI field for learning spatial filters for oscillatory processes.

To make optimal use of the algorithm, the following parameters have to be considered:

- Frequency band and time window are known
- band-passed signal is jointly Gaussian within the time window
- Brain patterns between two classes must differ

Theoretical background

For calculating the CSP filter-coefficients let

$$\underline{E}^{N\times T} \ldots \text{ single trial} \tag{2.1}$$

where N is the number of channels, T is the number of samples and C is the normalized Spatial Covariance matrix of E, noting in

$$\underline{C} = \frac{\underline{E}\underline{E}'}{trace\,(\underline{E}\underline{E}')}. \tag{2.2}$$

Now all available trials have to be separated by class, e.g. Hand MI and Feet MI, and averaged:

$$\underline{C}_H = \frac{1}{\#Hand} \cdot \sum_{i \in Hand} \underline{C}_i \qquad \text{respective} \qquad \underline{C}_F = \frac{1}{\#Feet} \cdot \sum_{i \in Feet} \underline{C}_i \tag{2.3}$$

A composite spatial covariance matrix C_c is created:

$$\underline{C}_C = \underline{C}_H + \underline{C}_F \tag{2.4}$$

Since a spatial covariance matrix is per definition positive semi-definite and a square matrix, the Eigenvalue decomposition of a matrix can be applied on C_c and factorized:

$$\underline{C}_C = \underline{U}_C \cdot \underline{\lambda}_C \cdot \underline{U}'_C \tag{2.5}$$

$$\underline{U}_C \ldots \text{eigenvectors of } \underline{C}_C$$
$$\underline{\lambda}_C \ldots \text{eigenvalues of } \underline{C}_C$$

Notice that $\underline{U}_C$ is the eigenvector matrix and $\underline{\lambda}_C$ is a diagonal matrix containing the eigenvalues. Now the matrices are reordered according to descending eigenvalues.

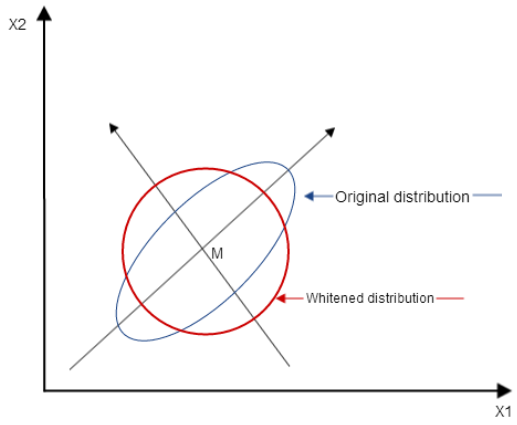


Figure 2.3: Whitening transformation: Equalizing the variances

To ensure proper scaling, a whitening transformation is applied (see schematic figure 2.3) which equalizes the variances in the space spanned by U_C,

$$\underline{P} = \underline{\lambda}_C^{\frac{-1}{2}} \underline{U}'_C \tag{2.6}$$

meaning that all eigenvalues of $\underline{P} \cdot \underline{C_C} \cdot \underline{P'}$ equal to one. By simultaneously transforming

$$\underline{S_H} = \underline{PC_H P'} \text{ and } \underline{S_F} = \underline{PC_F P'} \tag{2.7}$$

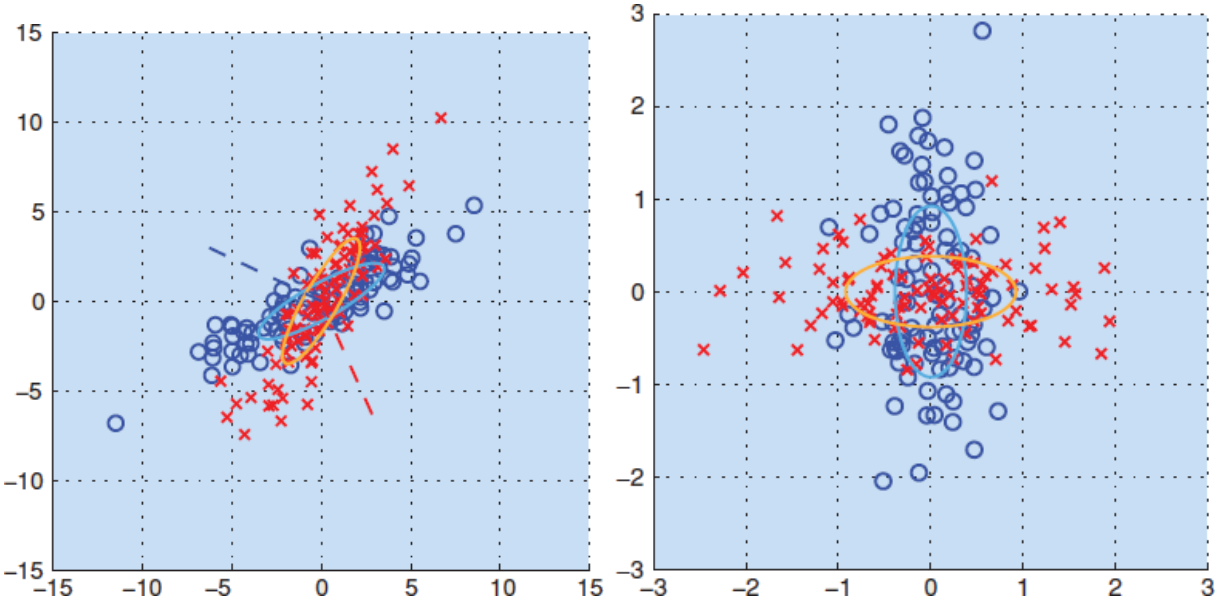
$\underline{S_H}$ and $\underline{S_F}$ share common eigenvectors $\underline{B}$ and due to the whitening/scaling their eigenvalue-matrix $\lambda_{H,F}$ add up to the identity matrix I.

$$\underline{S_H} = \underline{B\lambda_H B'} \text{ and } \underline{S_F} = \underline{B\lambda_F B'} \text{ where } \underline{\lambda_H} + \underline{\lambda_F} = \underline{I} \tag{2.8}$$

This leads to the fact that the sum of two corresponding eigenvalues is always one, so that the the eigenvalue with the largest value in S_H has the smallest value in S_F and vice versa. This property is fully utilized by the projection of the whitened EEG onto the the eigenvectors in B,

$$W = \left(\underline{B'P}\right)' \tag{2.9}$$

for feature vectors that are optimal for discriminating the two classes. The colums of W^{-1} represent the desired filter coefficients.



(a) Before CSP Filtering (b) After CSP Filtering

Figure 2.4: A toy example of CSP filtering in 2-D. Two sets of samples marked by red crosses and blue circles are drawn from two Gaussian distributions. In (a), the distribution of samples before filtering is shown.In (b), the distribution of samples after the filtering is shown. [22]

Applying CSP to the System

As mentioned before, for every filter output of the filterbank, all in all 15, one individual CSP-filter is applied. Since the most information, respective the highest variances, lie within the first and the last couple of vectors of the CSP filter W (see 2.8 and 2.9)the CSP filter is shrinked, leaving only the first and the last three columns resulting in a $13 \times 6, 13 \ldots$ #EEG Channels matrix which is applied to the filterbank signal $N \times 13, N \ldots$ samples.

$$W_{csp-matrix} = \begin{vmatrix} W1_H & W2_H & W3_H & W11_F & W12_F & W13_F \\ \vdots & \ddots & & & & \\ \vdots & & \ddots & & & \\ W1_H & W2_H & W3_H & W11_F & W12_F & W13_F \end{vmatrix}$$

Figure 2.5: Since the first and the last columns contain the most information for class separability, only the first and the last three colums of the original W - matrix are taken for further use.

2.2.3 Logarithmic Bandpower Features

Since the oscillatory changes of MI affect frequency and amplitude, the common approach of extracting the power information of the acquired signal seems valid.
Therefore, the output of the CSP - filtered signal is investigated for the power information from the signal and used as classification feature.

Incoming data samples are squared and a moving average filter over one seconds calculates the bandpower of the features (see figure 2.6). Afterwards the logarithm is applied.

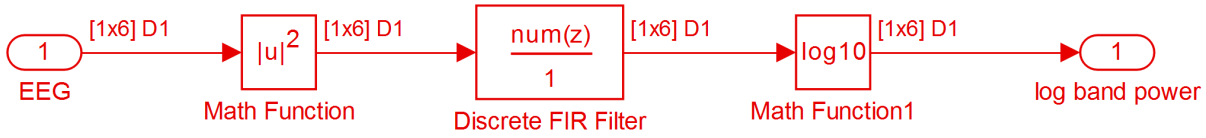


Figure 2.6: Extract from the Simulink Model: Calculation of the logarithmic bandpower features.

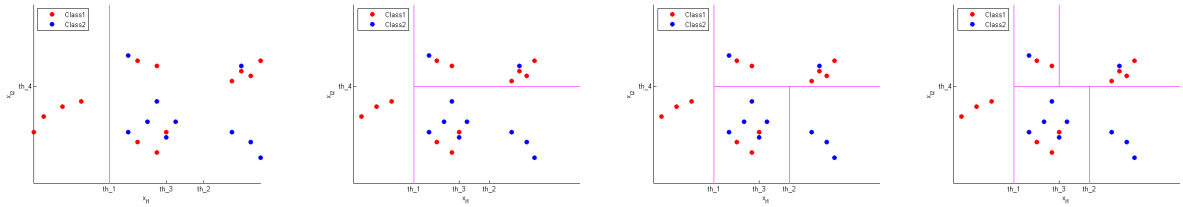
2.3 Classification: Random Forests

Random Forests were proposed by Leo Breiman in 2001 and are a further development of his previously proposed work on Decision Trees (CART = Classification and Regression Trees) [23] combined with his proceedings on Bagging Predictors [24] and Tim Kan Ho's work regarding Random Subspaces [25].
The technique is quite simple, but with state of the art performance. In 2006, a study conducted by Caruana et al. [26] investigated the performance of a number of supervised learning algorithms such as DT-based algorithms, Support Vector Machines (SVM), Naive Bayes and others on eleven prominent binary classification problems. The scores set decision-tree based classifier far to the front, with Random Forest on second place, only beaten by the approach of Boosted Trees.

2.3.1 Theoretical Background

Decision Trees

Decision Trees are the basis model of the Random Forest classifier. They are an extraordinary simple approach to the topic of classification and also regression, yet they can be quite powerful combined with additional techniques such as Bagging or Boosting. The main idea is to form a binary tree and minimize the error in each leaf of the tree. Therefore a sequence of binary splits is chosen of the data to divide it into different leafs, starting from the root node.



(a) first split (b) second split (c) third split (d) fourth split

Figure 2.7: Schematic of a possible feature space and the splits (pink lines) of a splitting criterion

Figure 2.7 shows a schematic of a possible feature space. The splitting thresholds split the feature space in rectangular regions. Figure 2.8 shows the growth of the decision tree. Starting from the root the first split separates the feature space in two regions (see figure 2.7(a)). The splitting criterion is based on a greedy heuristic and processed in an iterative way (see figures 2.7(b) - 2.7(d)).The final decision for the class is made by a majority vote of the leafs. In figure 2.8 the the process of building up the tree is shown in a detailed scheme.

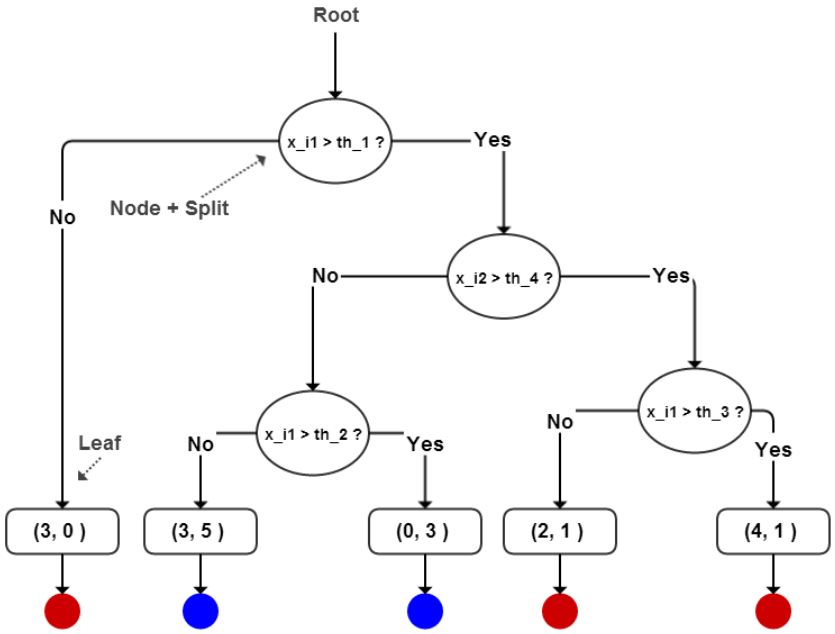


Figure 2.8: Schematic view of the growth of a tree using the feature space of figure 2.7

The main goal for a classification tree is to minimize the error in the leafs, meaning to get each region R as pure as possible. With each split the two consecutive regions should be "purer" than the single region they descended from. A common way to achieve this is to calculate an index for "purity" for each region, which can be perfectly used as splitting criterion.

Consider figure 2.9(a), where a node for classification is schematically represented. Let be

$$E_R = \text{ fraction of points } x_i \in R \text{ misclassified by a majority vote, then} \tag{2.10}$$

$$E_R = \frac{1}{N_R} \cdot \sum_{1:x_i \in R} I(y_i \neq Y) \tag{2.11}$$

where N_R is the total number of features in the observed node, R is the region and Y is the correct class label. In the case of the region represented in figure 2.9(a), $E_R = \frac{3}{8}$. For the next

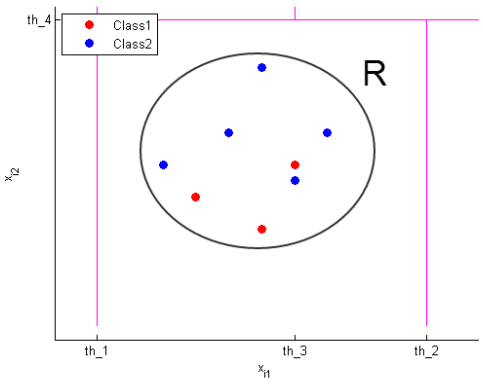
split of this node a threshold would have to be found which undercuts the MCR of this region. This purity measure is called Misclassification Error (MCE).
The second approach is to determine the entropy of the region, by

$$H_R = -\sum_{y \in Y} P_R(y) \cdot log(P_R(y)). \tag{2.12}$$

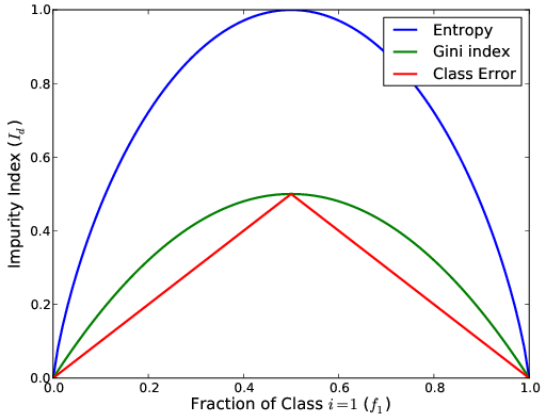
The last index approach presented is the Gini-Index, which will be later on used as splitting criterion by the random forests and is simply calculated by the sum of the probability of the occurrence of the class multiplied with its inverse probability for each class, so

$$G_R = \sum_{y \in Y} P_R(y)(1 - P_R(y)). \tag{2.13}$$

Figure 2.9(b) shows the three impurity indices as a function of the proportion.



(a) Schematic view of one splitted Region; Purity indices: $E_R = 0.375$; $H_R = 0.6616$; $G_R = 0.4688$



(b) Impurity measures as a function of the porportion

Randomisation Processes - Bootstrapping and Random Subspaces

Usually, single decision trees are not accurate enough to be used for classification problems, but combined with some other techniques, they can potentially become quite powerful to the task. Bootstrap Aggregation, often also called "Bagging", is another technique introduced by Leo Breiman in 1996 [27]. In principle, the method creates B number of instances of a decision tree, fabricating a so called ensemble classifier, and induces a majority vote over the results of each tree. However, this trick applied reclusively wouldn't lead to a significant improvement in accuracy. The general error would stay the same because each tree is build with an identical data set.

$$\text{Generalisation Error ... } PE \leqslant \frac{\overline{\rho}\left(1 - s^2\right)}{s^2} \tag{2.14}$$

Having a closer look on the formula of the generalisation error PE, two factors can be found. The accuracy s, which can not be modified in a reasonable way, and $\overline{\rho}$ which describes the mean value of the correlation of the base classifiers [27]. $\overline{\rho}$ is the factor which can be tuned by modifying the data set for each decision tree. Instead of giving each tree the full, identical training set

of data, Bootstrap Aggregation selects a randomly chosen set of training data ("get the samples out of the bag") for each set, with replacement. Therefore the diversity of the trees is greatly improved [27],[13].
To improve the diversity even further, the tree-growing process is modified by adding the Random Subspace method. When a region R, such as can be seen in figure 2.9(a) undergoes the process of finding the next splitting criterion, only a randomly chosen subset of features is selected to be considered for the splitting criterion [13].
Both randomisation processes applied to an ensemble of decision trees, called a forest, guarantee the diversity of the trees and optimize the generalisation error PE (see 2.14).

Assembling the classifier

The ideas of the decision trees and the randomisation processes are now combined into one single ensemble classifier producing the following algorithm:

1. Foreach Tree b = 1 to B ($B \ldots$ *number of trees*):
   (a) Draw a bootstrap sample Z of size N out of the training data (bootstrap step).
   (b) Grow a random forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each node until the minimum node size is reached.
      i. Select m variables of the p variables available in the region (random subspace step).
      ii. Pick the variable for the best split according to the Gini-Index.
      iii. Split the node in two daughter nodes.
2. Output the ensemble of trees $[T_b]_1^B$
3. Do a majority vote of the results of the trees and output the result as the classlabel.[27].

Parameters

As can be seen in the previous subsection there are several parameters of the Random Forest which can be tuned to improve the performance of the classifier. The first and obvious parameter is the number of trees the ensemble classifier should grow. The other parameter which offers the possibility of tuning is the number of features chosen by the second randomisation step for finding a good splitting criterion.
Fortunately, in 2012, Steyrl investigated in his Master Thesis the suitability of Random Forests for Brain Computer Interfaces, where he analysed of the tuning possibilities of the parameters, so these findings allow a good estimation of the parameters for using EEG data [14][13]. Therefore it is chosen to use 1000 trees to ensure on the one hand clear results of the majority vote of the trees, and on the other hand good computational efficiency allowing sixteen classifications per second. The number of features chosen by the random subspace method is set to $\sqrt{number\ of\ features}$.

2.4 System Model

The system model is based on the approach of Faller et al. Autocalibration and Recurrent Adaption BCI [10]. It is designed as a distributed system and consists of two main parts, which are realized in MATLAB Simulink and can be executed on two different machines (PC), or both

on the same PC using two instances of MATLAB. As can be seen in figure 2.9, the general BCI approach already seen in chapter 1, is extended by an Optimizer System which can be configured to retrain the CSP filter and the Random Forest classifier at pre-defined/recurrent timepoints to enhance their performance. The TCP/UDP/IP Toolbox 2.0.6 by Peter Rydesaeter is used, which allows communication of two MATLAB instances via TCP or UDP [28].

2.4.1 Software and Libraries

For the implementation, the following Software, tools and libraries have been used:

- MATLAB 2012b [29]
- Random Forest mex implementation for MATLAB [30]
- TCP/UDP/IP Toolbox 2.0.6 [28]
- Fast Serialize [31]
- TOBI SignalServer + Client [32], [33]
- GRAZ-BCI libraries [34]

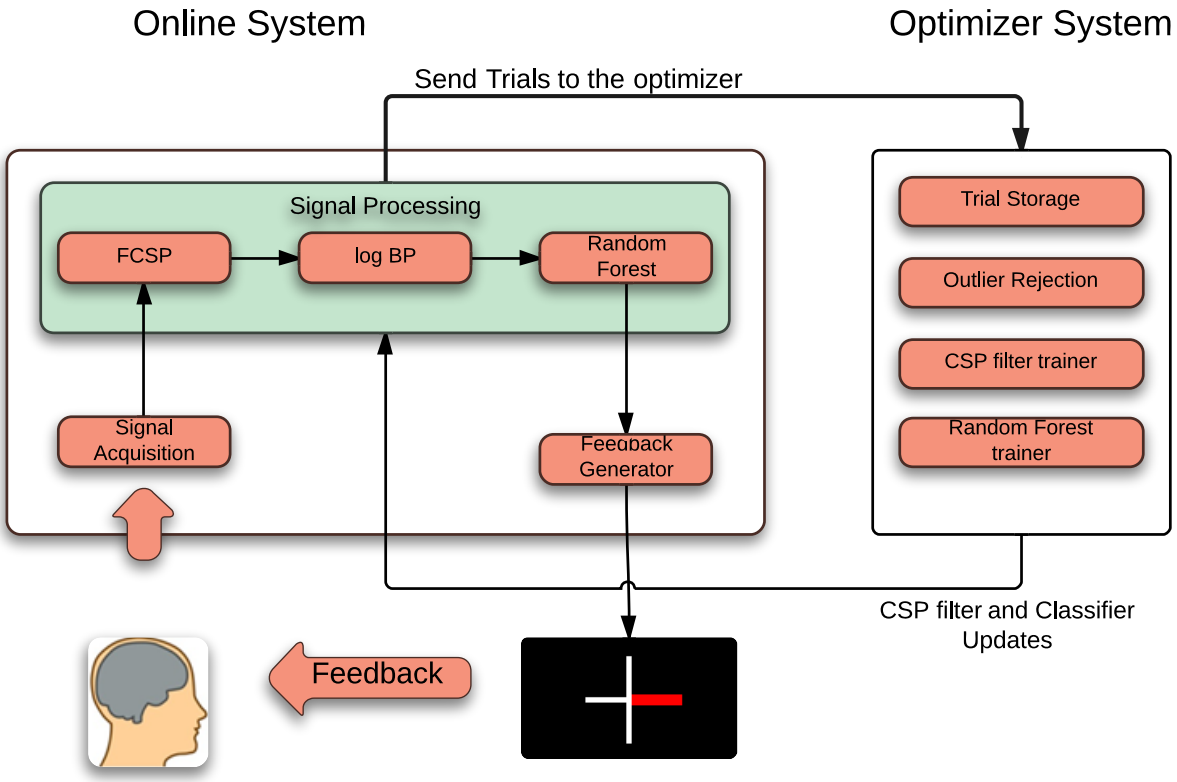


Figure 2.9: The general BCI approach is extended by a separate optimizer instance.

2.4.2 Onlinessystem

The Online System itself can be almost directly derived from the standard BCI system commonly known. It is implemented in MATLAB Simulink, using the Tobi Signalserver and the libraries for the GRAZ-BCI. Detailed Simulink Models as well as critical code compartments are provided in the Appendix.
A full schematic can be seen in figure 2.10. The system is designed to operate with thirteen input channels (which represent a setup of thirteen electrodes), but can be effectively scaled to work with other configurations. The Online System is realized in Simulink.

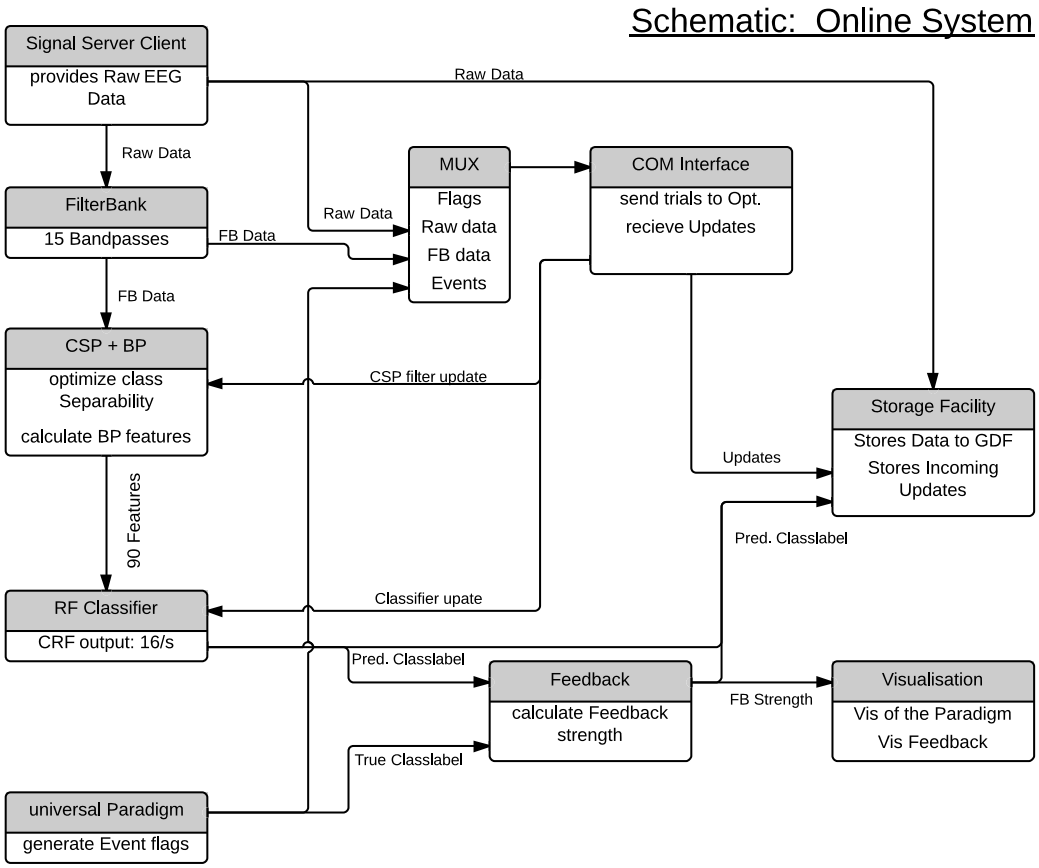


Figure 2.10: Detailed Schematic of the Online System

The acquired signal is provided by the Signal Server client [32], processed in real time and directed to the filterbank where the signal is fed in fifteen independent bandpass filters (Filterbank). By processing the signal with fifteen parallel filters, the originally thirteen input channels multiply to 195, thirteen for each filter. Each filter output is modified by multiplication with the common spatial pattern (CSP) filter, which is trained individually for each filter. The output results in six bandpower features for each filtered CSP. Adding these up results in a total of ninety features.
The Random Forest Classifier classifies the CSP+BP output sixteen times per second and therefore delivers 16 predicted classlabels per second. This is an applicable trade-off between classification accuracy and computational performance.

Another critical object in the model is the universal Paradigm generator. It provides events allowing trigger points for the visualisation and the feedback model, as well as the true class label. The universal Paradigm generator is part of the GRAZ-BCI library and is driven by a XML file, which can be adapted to provide triggerpoints, time, order and succession.
To ensure concurrent processing between different sample rates (e.g. the basis sample rate for the model is 256, while the output of the classifier delivers only 16 decisions in the same time), various rate transitions are placed on critical points (which are not depicted in figure 2.10).
The true class label as well as the predicted class label drive the feedback generator which calculates the "strength" of the feedback.
The visualisation block handels the visualisation and is driven by the universal Paradigm which triggers the different states of the Paradigm and the feedback block, which determines the strength of the feedback bar.

The raw signal, the output of the filterbank as well as the trigger events are collected in the MUX for each timestep and forwared to the COM Interface.
The COM interface is critical for the entire system. It maintains the communication to the previously addressed Opitimizer System, allowing on the one hand classifier and filter updates for the CSP, and on the other hand segments and sends trials to the Optimizer System. Segmenting is the technique of selecting periods of training data, which are defined by the paradigm selected. Each trial begins with a starting event and has the same length. The COM interface monitors the incoming data stream from the MUX for such events. When the event occurs, it saves the incoming data to a buffer for the size of $M \times \textit{Trial length} \cdot \textit{Sample Rate}$. After filling up the buffer, the package is sent to the Optimizer System via TCP/IP. A schematic view of the COM interface can be seen in figure 2.11.

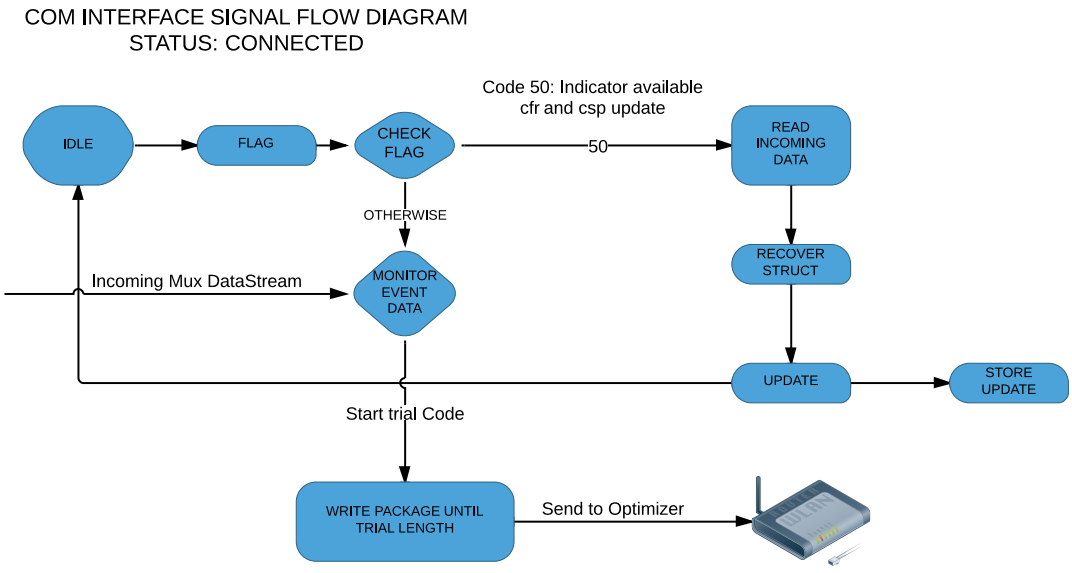


Figure 2.11: The COM-Interface: Responsible for handling incoming updates and segmenting trial according to the presets given by the paradigm

The storage facility is liable for the storage of the data for later offline analysis. Everything non-reproducible is stored in the designated GDF Data file, or in MATLAB files including:

- Raw EEG Data (GDF)
- True Classlabel (GDF)
- Predicted Classlabel (GDF)
- System to Sim Time (for monitoring eventual delays, GDF)
- Each Classifier and CSP Filter update (MATLAB file)

2.4.3 Optimizer System

The Optimizer System is realized as a MATLAB function, and can be executed on the same PC as the Online system or on another PC. The pre-requisite is that both computer are connected via TCP/IP. The Optimizer System acts as a "always-on" server during a session. The Online system connects and disconnects to the system according to the number of runs planned for the session implying that the data collected in previous runs is available for the present and future runs. This is actually one critical feature in order to calculate "optimized" updates for the Online system. There are three main assignments the Optimzer performs:

1. Data management and segment storage
2. Outlier Rejection (Artefact detection)
3. Training of the CSP Filter and the Random Forest Classifier

Similar to the COM Interface of the Online System, the Optimizer operates in states using flags as triggers. Figure 2.13 displays a basic signal flow chart of the optimizer in the connected state. It is assumed that the Online System has already connected to the Optimizer and is sending Data-Packages (which could be trial segments or global commands) to the Optimizer. Each Data-Package contains a flag, which is the indicator on how it is handled in the further process. Flag zero shuts down the Optimizer and can be sent manually (usually only needed when the recording session with the subject is concluded). Flag one indicates the end of a current run resulting in closing the actual connection to the Optimizer. Each time the Online System is started, it opens a new connection to the Optimizer independently.
Flag two reports the arrival of a new trial.

Data management and trial storage

Every new arriving trial is stored in the data matrix and labelled according to its class label. Since the trial is segmented and rid out of the real time data flow of the online system, the previous trigger points of the universal Paradigm generator mismatch but new can be constructed since length and gaps between events are perfectly known and therefore no shifts or data loss happens.
The output of the optimizer strongly relies on the pre configuration of a selection of variables which define when updating starts or the number of updates:
With reference to the offline tests in [28] the number of initial training starts is set to ten trials per class. Consecutive updates happens every four new trials per class. Keeping the number of classes used for retraining equally distributed (although it is not a prerequisite in any form for classifier training) allows to compare the performance of the system with similar e.g. Faller 2012 [10].

Outlier Rejection (Artefact detection)

The EEG signal itself is highly vulnerable to a broad variety of noise signals, called artefacts. These artefacts divide either to non physiological sources, such as power-line noise (50/60Hz) and moving electrodes, or physiological sources such as potentials induced by eye-movement, blinking or muscle movement.[6] While the influence of non-physiological sources can be reduced to a minimum by proper preparation of the equipment, physiological potentials induced by eye-movement and blinking , or muscle movement are hard to come by. To illustrate the effect of artefacts, figure 2.12 shows a variety of common physiological artefacts recorded during a run. Plot (A) shows a high amplitude pattern caused by multiple blinks. Due to the position (FP1, frontal area) of the electrode the artefact evolves to its full extend. Plot (B) shows the effect of eye movement such as "looking around" or "eye-rolling" (low frequency patterns). Plot (C) lists a muscular artefact (EMG) caused by head movement, (D) shows the effect of teeth grinding (EMG).
Ignoring or not dealing with artefacts in a proper way may result in severe consequences, since these artefacts modify the shape of the signal (as can be seen in figure 2.12), they take direct influence on the input of the BCI system and may - in worst case, drive the BCI system itself rather than actual brain signals [35]. So by not dealing with artefacts in a proper manner the most advanced and sophisticated classification algorithm may be trained instead of brain signals, on an unconsciously performed nose itching or eye-rolling.

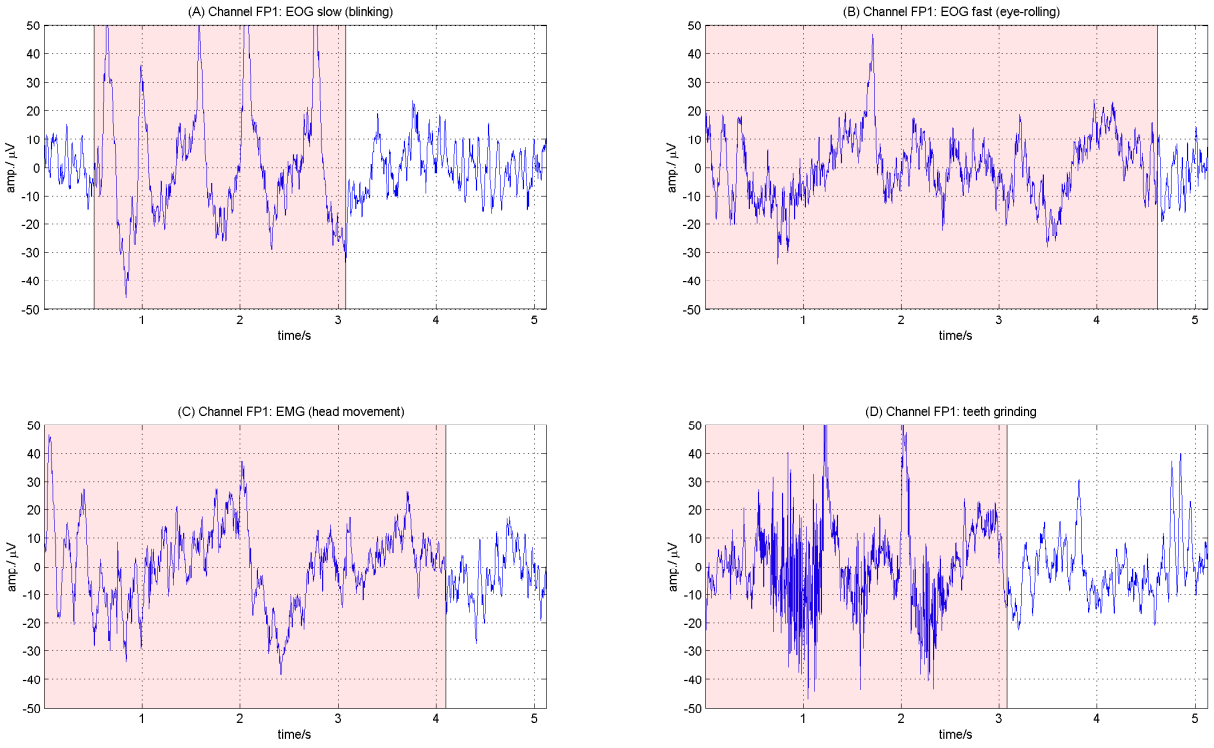


Figure 2.12: A selection of artefacts in EEG, subject CC02, bandpass-filtered[1,40]; f.l.t.r: (A) EOG: blinking; (B) EOG, eye-rolling; (C) EMG, head-movement; (D) EMG, teeth-grinding

Therefore, a statistical Outlier Rejection is implemented to counter the effects of artefacts by excluding every trial which is flagged to be contaminated. The described Outlier Rejection Method is based on the EEG lab toolbox by Delorme et al. [36], and is used similarly by Faller 2012 [10]. For detecting artefacts, the following statistical methods have been applied:

a.) **Rejection by amplitude threshold:** Standard thresholding is a quite simple and common

way to detect contaminated trials. The trial is rejected if the value of any data point in the trial exceeded the threshold. Thresholds are set to $\pm 100 \mu V$.

b.) **Rejection by channel variance:** Thresholding by variance of each channel. If the threshold is higher than 5 times the standard variance, the trial is rejected.

c.) **Rejection by probability:** Most of the artefacts (as can be seen in 2.12) have "unusual" behaviour over time which can be found by using the joint probability of the values of the trial in one column and compared to the probability distribution of all columns.

d.) **Rejection by kurtosis:** For rejection of unusual probability distributions the kurtosis of the trial can be determined and compared to a threshold. [37]

As depicted in 2.9, after an arriving trial is stored the Optimizer checks the available data for the number of trials related to each class. If the conditions for the minimum trials for an update are fulfilled, the stored Raw data is passed to the Outlier rejection method for artefact screening. Trials assumed to be tainted are excluded from further processing and the Optimizer checks the conditions for an update again. If all criteria are met, the training can be executed, otherwise the Optimizer awaits the next trial.

CSP training + Random Forest

For training the CSP filters the **Filterbank** data is used. The trials are split up according to their class labels. Each trial is segmented so that the period from second 4.75 to 7.75 is selected for training. The resulting filter is squeezed, so that only the first three and the last three channels, which are supposed to contain maximum class separation information, remain. For further processing and training the Random Forest classifier, features of the available trials are calculated. From each trial the features present at second 5.5 are taken for retraining the classifier. This time point is chosen because the most discriminative components can be found there on average (see [38]) when operating a BCI using SMR.

Packaging and Sending

Because of the complex data structure of the random forest classifier, serialisation of the data [31] has to be done in order to send it via TCP/IP [28]. CSP filters and the new classifier are serialized and sent to the Online system, where the COM Interface applies the updates and stores them separately for further analysis (see figure 2.11).

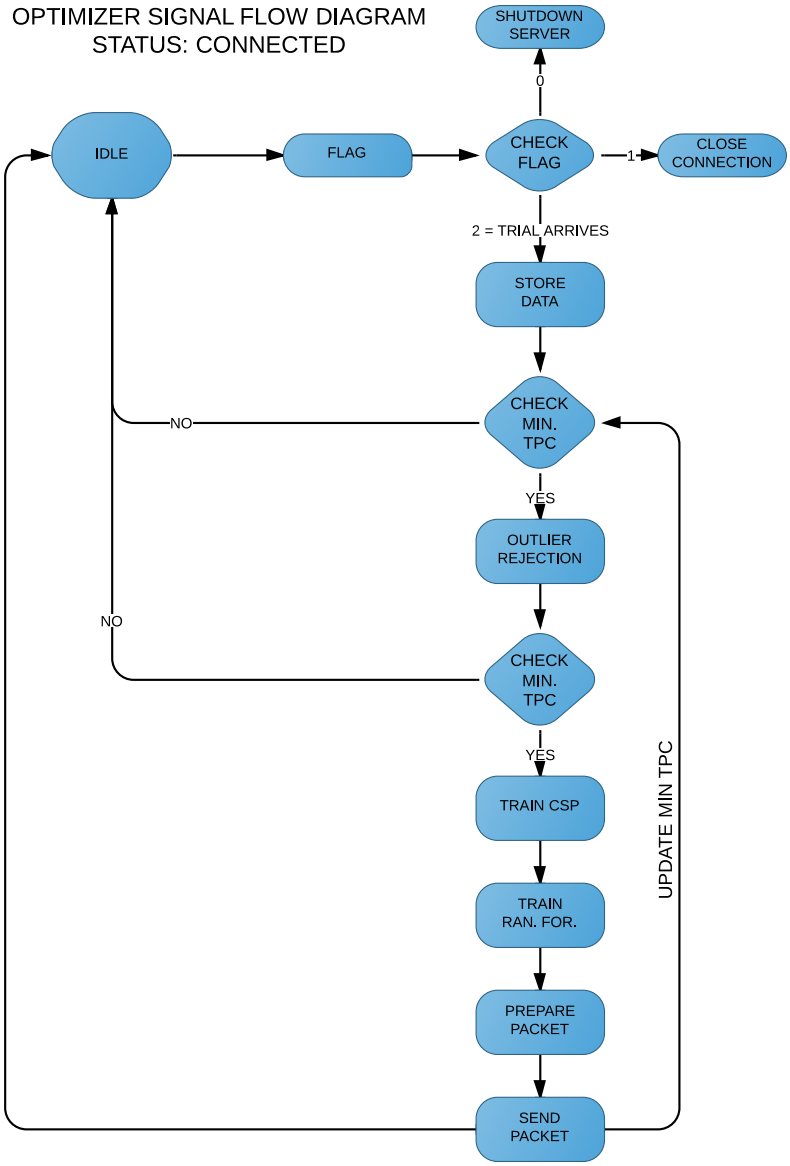


Figure 2.13: Optimizer Signal Flow Diagram in connected state.

2.5 Paradigm and Feedback

The paradigm, which essentially describes the composition of a trial consists of an 8 second activity period followed by a pause of random length, at least 2 seconds but not longer than 3. The Paradigm is displayed in figure 2.14:

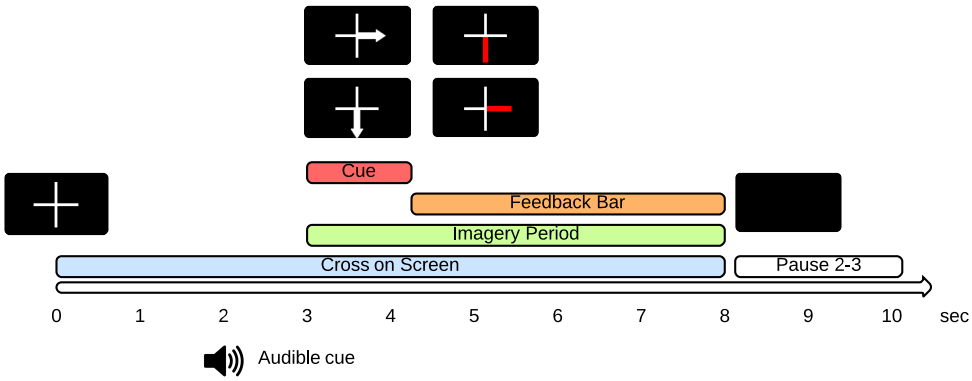
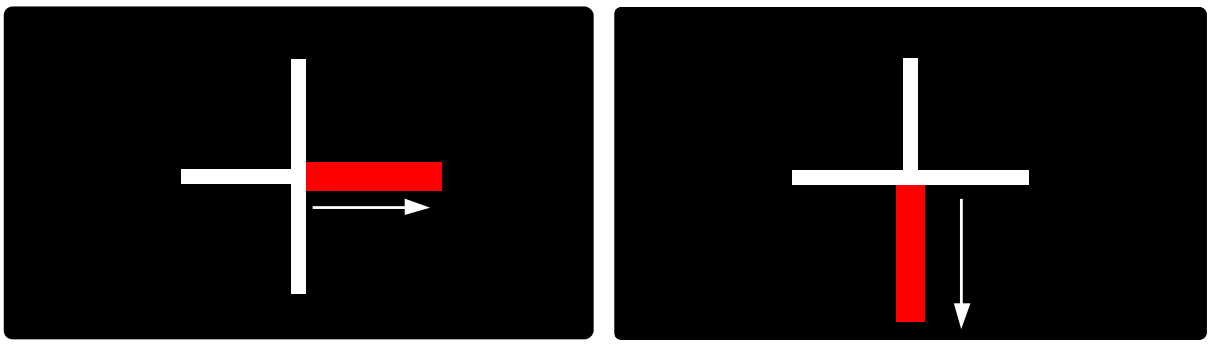


Figure 2.14: Optimizer Signal Flow Diagram in connected state.

- **Second 0:** A white cross appears on the screen. The cross stays on screen for the whole trial.
- **Second 2:** An audible cue (Beep!) is played to get the subjects attention.
- **Second 3:** An arrow is displayed on the screen. The arrow randomly points to the right, indicating the motor imagery of the right hand, or down indicating the motor imagery of both feet. The subject is instructed to start with the motor imagery as soon as an arrow appears on the screen.
- **Second 4.25:** The arrow disappears. If the number of minimum trials per class has already been reached, the feedback bar-graph appears growing to the right for successful detection of motor imagery of the right hand, and growing down for successful detection of motor imagery of the feet.
- **Second 8:** Feedback bar and cross disappear and the pause begins.



(a) Feedback bar grows to the right

(b) Feedback bar grows down

Figure 2.15: Feedback bars

Feedback is given by a red graph-bar who grows in the direction the cue was pointing before. The length of the bar is normalized by the number of correct classifications in the past second.

2.6 Experiment Setup

The experiment took place in a controlled laboratory environment. Subject measurement happened in a shielded box (see 2.16(a)), where the participant was seated comfortably in a leather chair. The Paradigm was displayed on a screen which was positioned approximately one meter in front of the subject. For the audible cues, speakers were placed next to the screen (see 2.16(b)). For safety and surveillance of the behaviour of the subject, a camera was mounted in the box.

(a) Measurement Box (b) Inside the measurement box

Figure 2.16: The experiment took place in a reproducible controlled environment

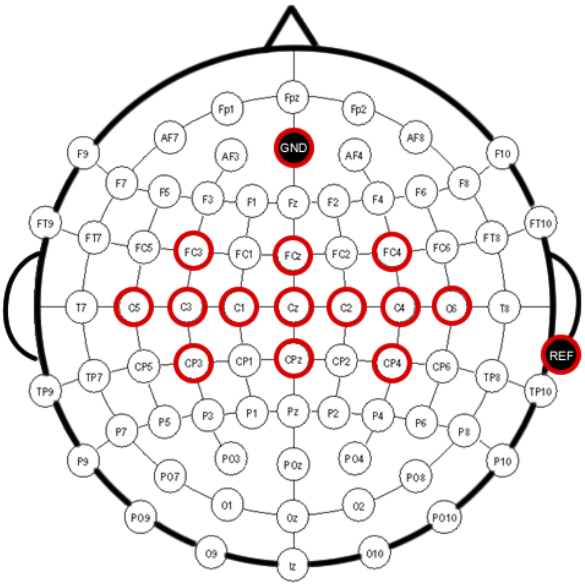
For data acquisition the active electrode system g.GAMMAsys by g.tec is used, as well as g.USBamp [39]. The electrode cap is fit with 13 active electrodes which are placed according to the international ten-twenty system [40]. Table 2.17 show the exact electrode positions as well as figure 2.17(a).

The output of the electrode system was amplified by a g.USBAmp, which was connected to the Signal Server. The data itself was recorded with a sampling rate of 256 Hz. Intrinsic methods of the g.USBamp [33] allowed prefiltering: A 8th order chebyshev bandpass-filter in the range of 0.5 to 100 Hz was applied, as well as a notch filter with the center at 50Hz. The chebyshev filter restrains the frequency range, while the notch filter explicitly covers the mains frequency (50Hz power-line noise).

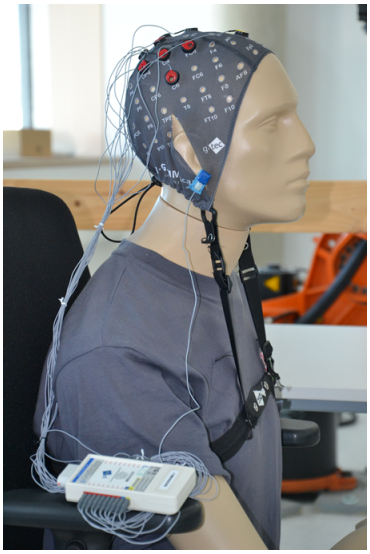
Electrode	1	2	3	4	5	6	7	8	9	10	11	12	13	GND	Ref
	FC3	FCz	FC4	C5	C3	C1	Cz	C2	C4	C6	CP3	CPz	CP4	AFz	r. Ear lobe

Table 2.2: Electrodes used according to the international ten-twenty system [40]

The electrode cap is fixed on the subjects head by using a chest band instead of a usual chin band, see 2.17(b). This method is more comfortable for the subject, the advantage is that the EMG influence of the masticatory apparatus decreases on the EEG.
Instructions for motor imagery are the same for each subject: For Hand MI, the subject should imagine squeezing a training ball in a continuous way using the right hand, for FEET MI the subject should imagine pressing its feet iteratively on a box.



(a) Schematic of the electrode positions



(b) Electrode cap and g.gammaSYS rigged on a dummy. The electrode cap is fixed using a chest band.

Figure 2.17: Electrode Positions and Cap fixation demonstrated on a dummy.

2.6.1 Recording Session characteristics

Each recording session lasted no longer than 90 minutes. Preparation of the subject took about 10-15 minutes, detailed explanation of the paradigm and the instructions another fifteen minutes. The session consisted of 4 runs with 20 pseudo-randomized trials per class (right hand, both feet) . After the first 10 trials per class which passed the outlier rejection, the Random Forest classifier and the CSP filter were trained and the system started to give feedback to the subject. On average, first feedback to the volunteer could be given after 4 minutes. The consecutive runs got a classifier update at the start of each run, therefore the subject got full feedback after the first training. The further updates happened after 4 new trials per class passed the Outlier Rejection.

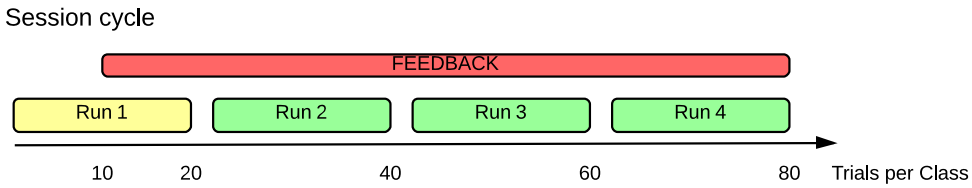


Figure 2.18: Session Cycle. After each run a short checkup was done to ensure the subjects comfortness and fitness.

3 Results

3.1 Preliminary tests using non naïve subjects

minimum accuracy for exceeding chance level		
trials per class	chance level [%]	alpha
70	59.693	> 0.01

Table 3.1: Minimum accuracy for exceeding chance level

To show functionality and first performance tests, preliminary tests with three non-naïve subjects were executed. These subjects used at least once in their lifetime a SMR based BCI. Figure 3.1 shows the mean accuracy for each subject calculated over the trial period. To overcome chance level the achieved accuracy had to be higher than 59.7% (see table 3.1). The blue perpendicular line indicates the time-point of the onset the cue. Table 3.2 lists the peak accuracies as well as the calculated mean and median over the feedback period. The non-naïve subjects were able to reach and average peak accuracy of $91.42 \pm 8.66\%$

Figure 3.2 shows the detection rate of each class as well as the mean accuracy for each subject.

Preliminiary Tests: Right Hand versus Foot				
subject	peak [%]	peak reached at [s]	mean(4.5-7.5s) [%]	median(4.5-7.5s) [%]
CC02	92.86	5.93	84.36	85.00
BE02	99.29	4.96	96.60	97.86
CV06	82.14	5.34	76.84	77.86

Table 3.2: Preliminary tests: Evaluation Right Hand vs Both Feet

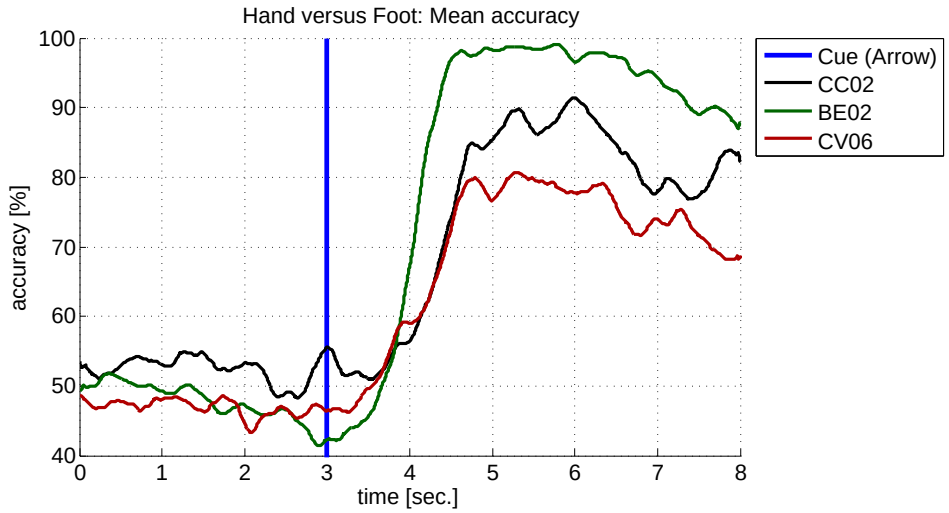
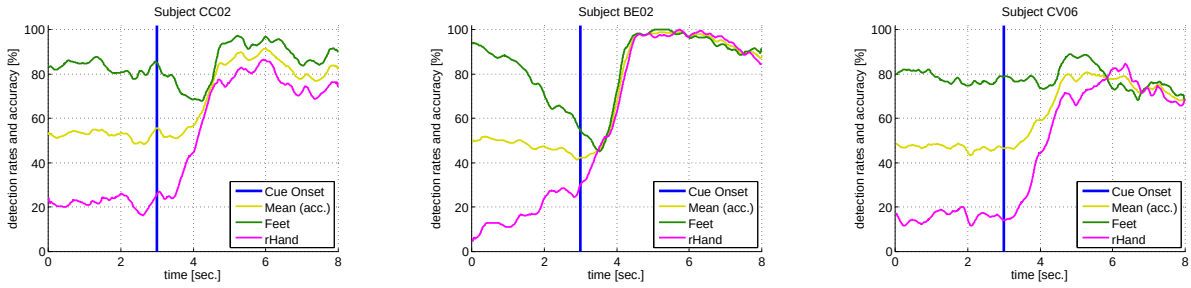


Figure 3.1: Results of the Preliminary tests: Mean accuracy over the trials right hand versus foot. The perpendicular blue line represents the onset for the cue.



(a) CC02 (b) BE02 (c) CV06

Figure 3.2: Preliminary Tests, Detection rates: Each plot contain the probability information of right hand (pink), both feet (green) and the mean of both classes (accuracy) in gold.

3.2 Performance of naïve subjects

3.2.1 Mean over both classes

For evaluating the performance of the system , 9 novice, healthy volunteers, 8 males, 1 female, from the age of twenty to thirty were measured. Each subject underwent one single session which was done uninterrupted. The duration of the session was - including preparation of the subject- not longer than 90 minutes.

The basis of figures 3.2,3.3, 3.4 and 3.5 is the activity period of eight seconds in each trial. Figure 3.3 shows the mean accuracy of both classes over all trials. The perpendicular blue line at second three represents the onset of the cue (arrow right or arrow down, pseudo-randomised) indicating the start of the motor imagery period. Average peak performance levelled in at $84.84 \pm 10.27\%$.

Table 3.3 summarizes the calculated accuracies as well as mean and median accuracy over second 4.5 to 7.5 seconds for each individual subject as well as the mean of peak mean and median.

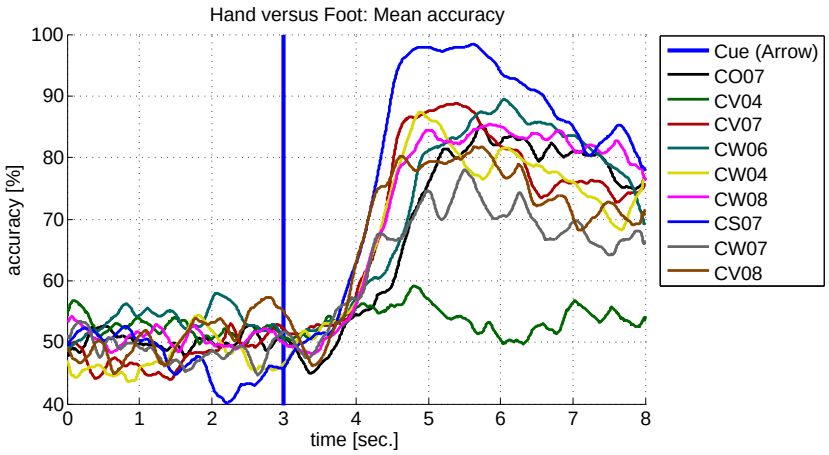


Figure 3.3: Result Plot Mean accuracy over the trials Left versus foot. The blue bar represents the timepoint of the onset of the cue.

Hand versus Foot: True label versus Predicted label

subject	peak [%]	peak reached at [s]	mean(4.5-7.5s) [%]	median(4.5-7.5s) [%]
CO07	86.43	5.70	79.24	80.71
CV04	61.43	4.80	53.84	53.57
CV07	89.29	5.20	81.62	81.43
CW06	90.71	6.06	82.54	83.57
CW04	88.57	4.85	79.26	78.57
CW08	86.43	5.98	82.75	82.86
CS07	98.57	5.57	92.18	93.57
CW07	79.29	5.47	70.44	70.00
CV08	82.86	5.55	76.43	77.86
Averages ± Std.	84.84 ± 10.27	5.46 ± 0.45	77.59 ± 10.62	78.01 ± 11.05

Table 3.3: Evaluation Right Hand vs Both Feet

3.2.2 Detection rates for Right Hand and both Feet Motor Imagery

The detection rate for the class hand is shown in table 3.4 as well as individual peak mean and median. Two subjects show the peak of the detection clearly before even the cue happened (signed with an asterisk). This indicates a bias of the classifier towards that particular class.

Class hand MI: True label versus Predicted label

subject	peak [%]	peak reached at [s]	mean(4.5-7.5s) [%]	median(4.5-7.5s) [%]
CO07	85.71	5.70	76.83	80.00
CV04	72.86	0.93*	53.09	51.43
CV07	94.29	4.70	80.62	78.57
CW06	91.43	5.61	80.88	82.86
CW04	84.29	4.85	70.45	70.00
CW08	90.00	4.95	83.60	84.29
CS07	98.57	4.78	93.19	94.29
CW07	81.43	5.47	71.68	71.43
CV08	87.14	0.45*	78.17	78.57
Averages ± Std.	87.30 ± 7.54		76.50 ± 11.05	76.83 ± 11.90

Table 3.4: Evaluation Class Hand MI

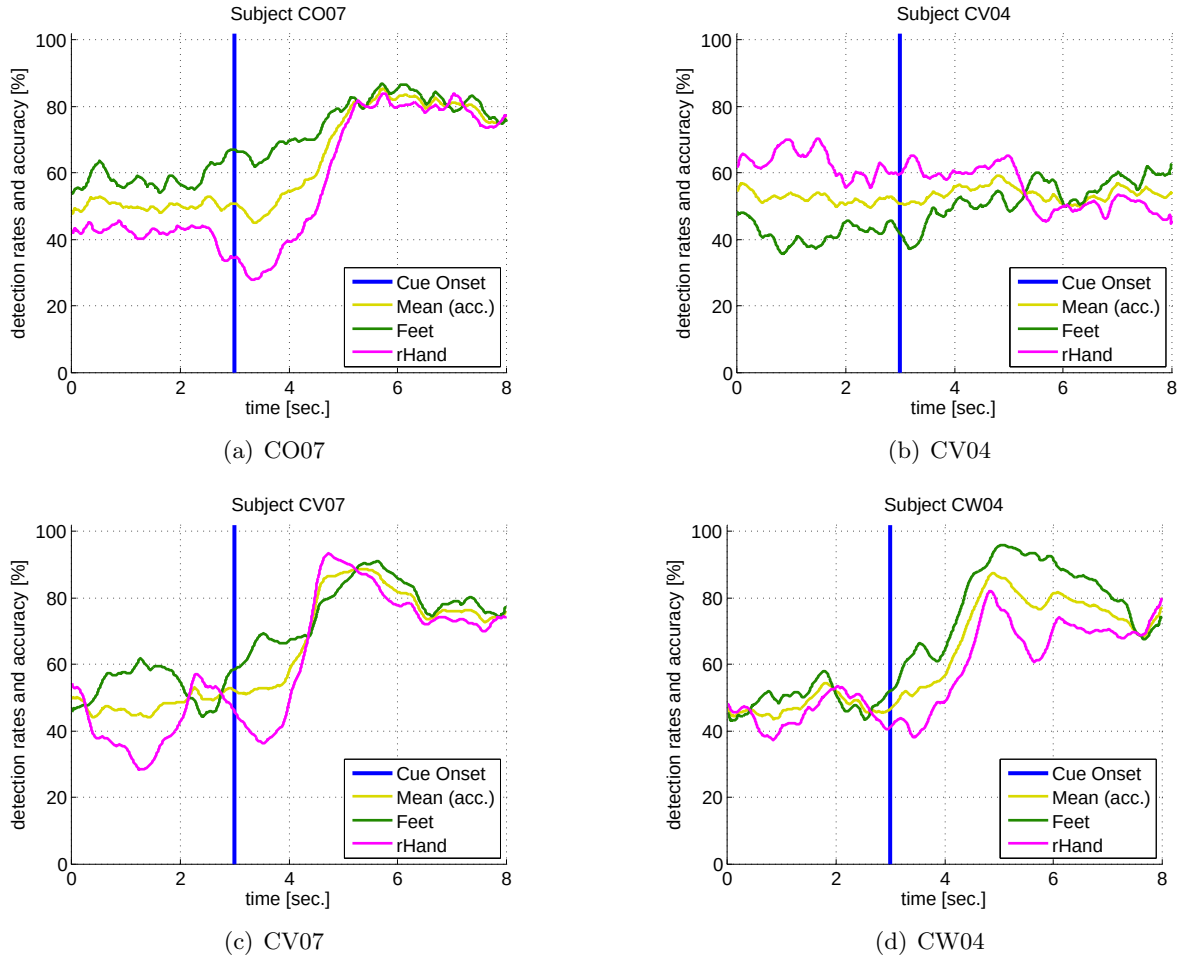
The detection rate for class feet is shown in table 3.5, as well as the individual peak, mean and median values.

Class FEET MI: True label versus Predicted label

subject	peak [%]	peak reached at [s]	mean(4.5-7.5s) [%]	median(4.5-7.5s) [%]
CO07	88.57	5.68	81.65	81.43
CV04	64.29	7.54	54.60	54.29
CV07	91.43	5.34	82.62	81.43
CW06	94.29	6.22	84.20	85.71
CW04	97.14	5.20	88.07	88.57
CW08	90.00	5.78	81.90	82.86
CS07	98.57	5.16	91.18	92.86
CW07	78.57	5.45	69.19	70.00
CV08	88.57	5.70	74.68	77.14
Averages ± Std.	87.93 ± 10.62		78.77 ± 11.16	79.37 ± 11.45

Table 3.5: Evaluation Class Foot MI

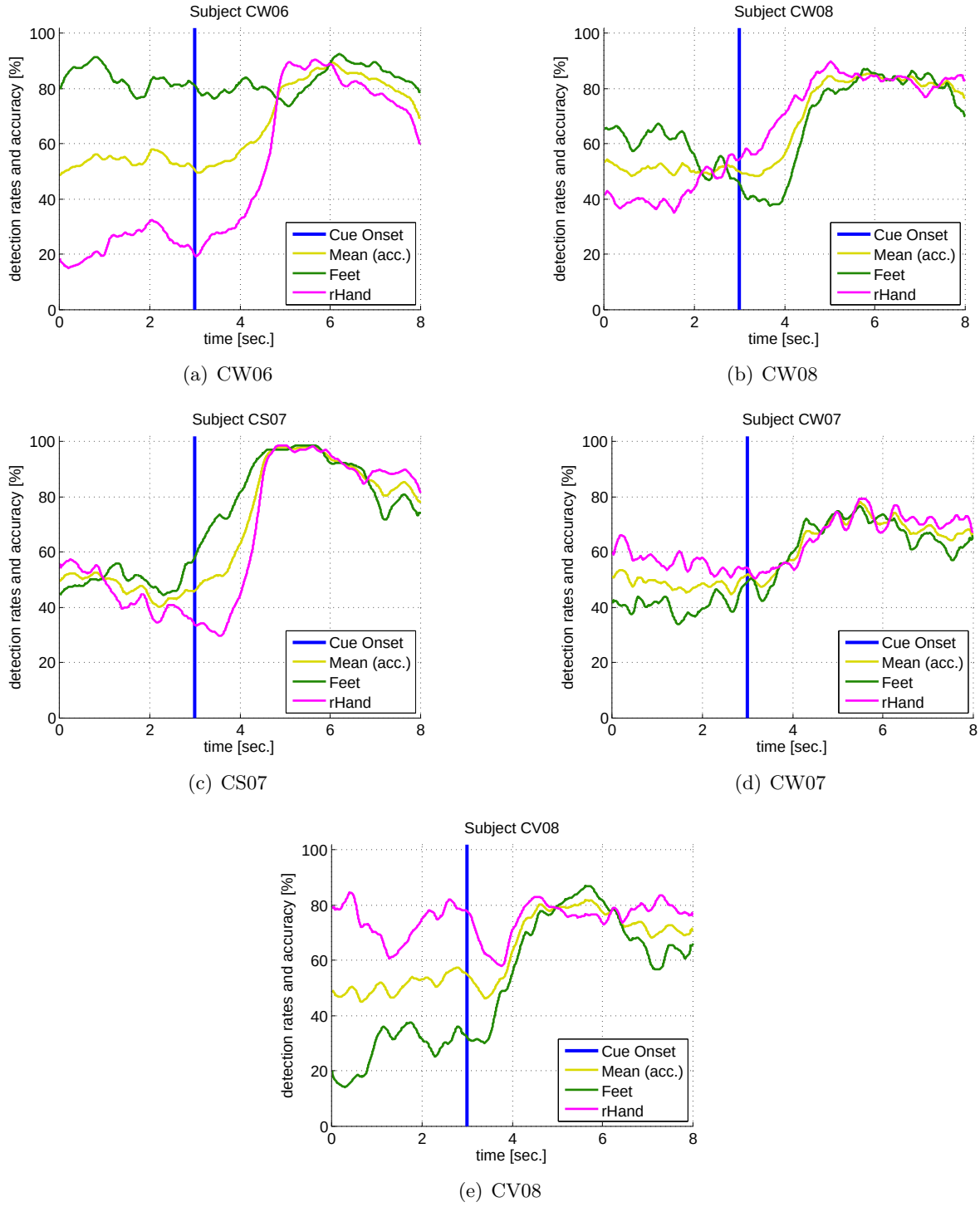
Figures 3.4 and 3.5 show the detection rates of both classes as well as the accuracy over both classes plotted over the trial time for each subject. The blue perpendicular line represents the onset of the cue.



(a) CO07 (b) CV04

(c) CV07 (d) CW04

Figure 3.4: Detection rates and accuracy plots for each subject: Each plot contains the detection rates of right hand (pink), both feet (green) and the mean of both classes (accuracy) in gold.



(a) CW06

(b) CW08

(c) CS07

(d) CW07

(e) CV08

Figure 3.5: Detection rates and accuracy plots for each subject: Each plot contains the detection rates of right hand (pink), both feet (green) and the mean of both classes (accuracy) in gold.

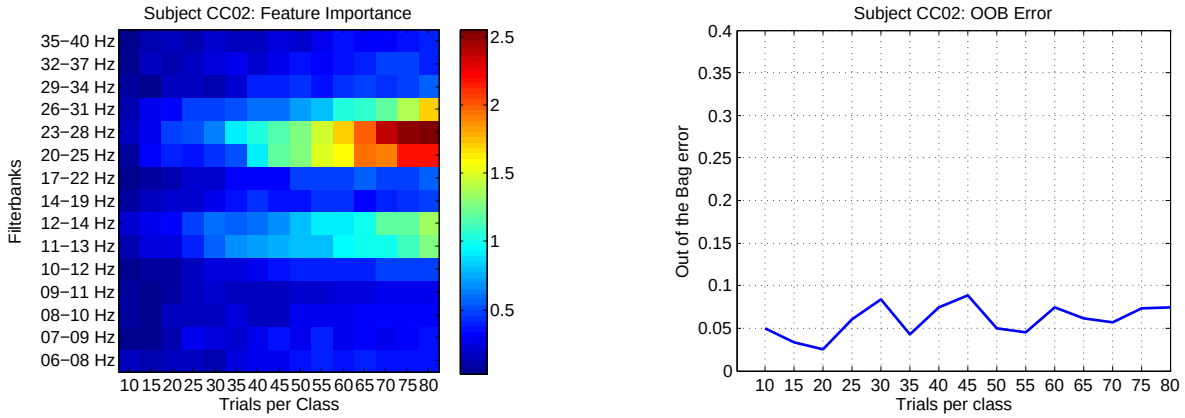
3.3 Classifier Statistics

A quite beneficial advantage of the used Random Forest implementation are the intrinsic methods for analysis [30]. One of these methods can be used to determine the importance of each feature used for building up the decision trees. The scale for the importance of a feature is the mean decrease of the Gini index.

Since multiple classifiers are trained with increasing number of trials per class a trend for the importance of each feature can be displayed. The results are displayed in the colormap as can be seen in figure 3.7 ff. (a.) .

The number of training points differs per subject. This is the result of the Outlier rejection: artefact-contaminated trials are dropped.
The second method which was used for evaluation is the "Out of bag" error estimate (OOB), which is the estimator of the test set error. For each training sample the decision trees of the classifier are investigated for the occurrence of this training sample. Those, where the sample did not occur are selected to build up a new random forest classifier(sub-classifier).A classification with the sub-classifier(which has fewer decision trees than the its parent) for the sample happens and the accuracy can be calculated.
This process is similar to a k-fold cross validation [27], apart from the fact that the for each sample investigated the number of trees the sub-classifier changes. Therefore it is called an "error-estimate". The figures 3.7(b) ff. display the OOB-estimate over the update-steps for each subject.

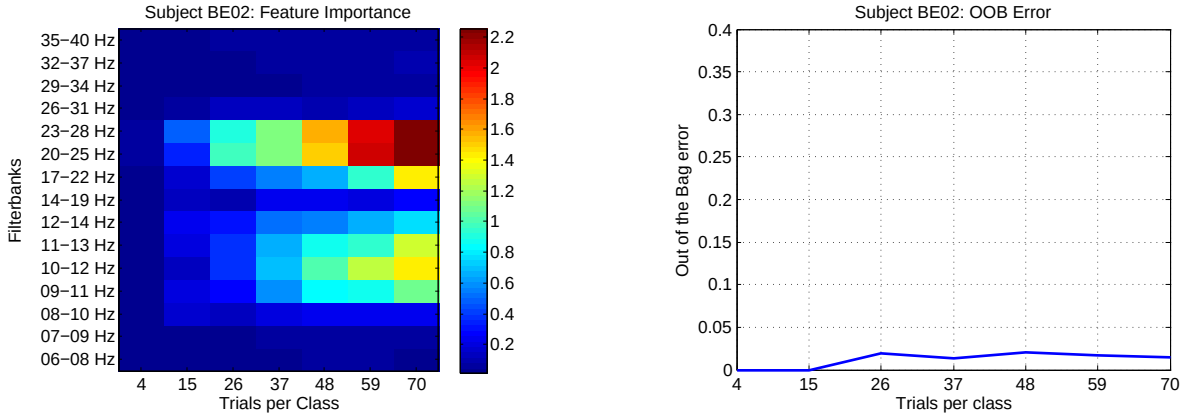
3.3.1 Preliminiary tests: non naïv subjects



(a) Feature Importance

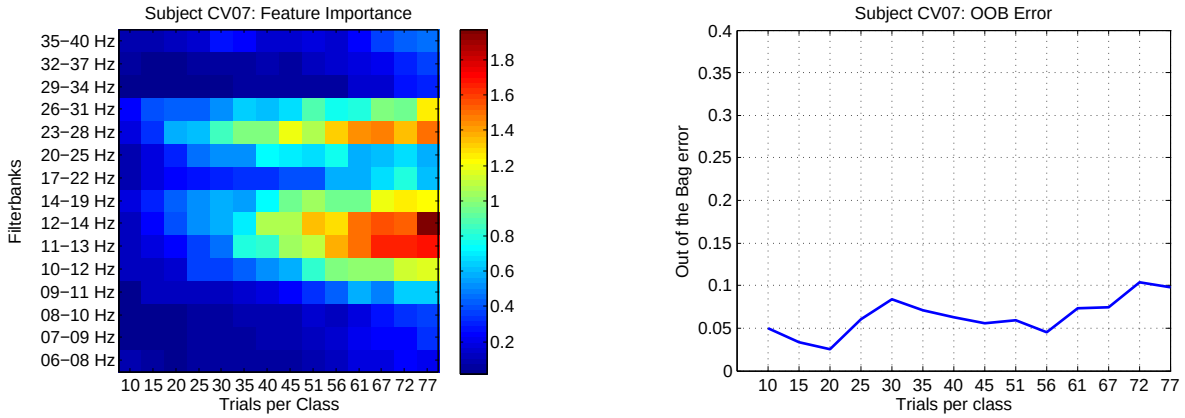
(b) OOB-Estimate

Figure 3.6: CC02: Feature importance and OOB-Estimate



(a) Feature Importance (b) OOB-Estimate

Figure 3.7: BE02: Feature importance and OOB-Estimate

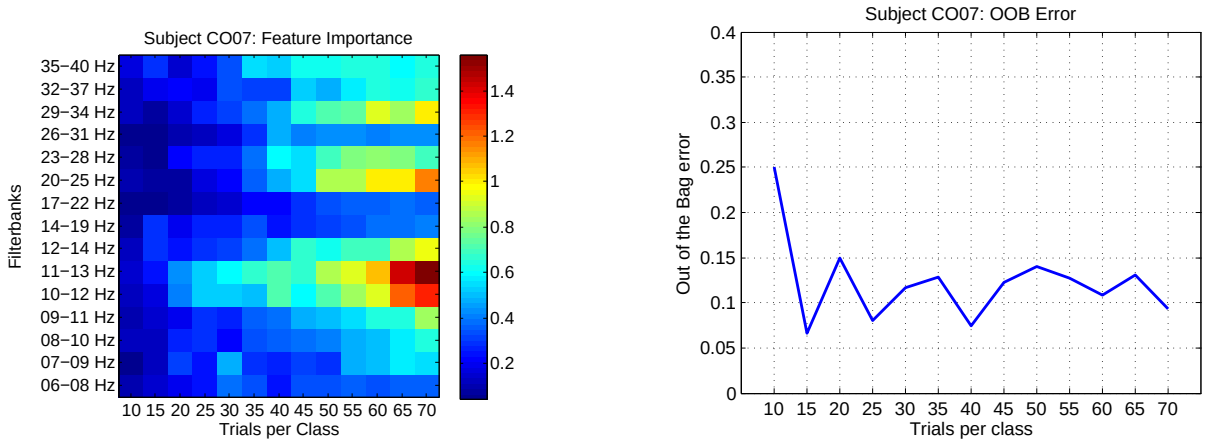


(a) Feature Importance (b) OOB-Estimate

Figure 3.8: CV07: Feature importance and OOB-Estimate

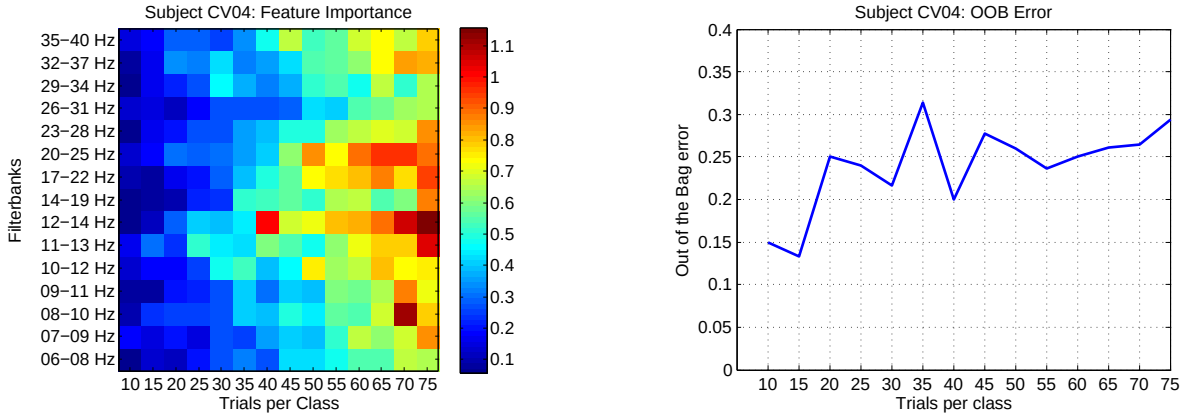
3.3.2 Naïve subjects

To investigate how well the out of the bag error estimation performs, table 3.6 shows a comparison between the overall peak accuracy after 70 trials per class and the OOB estimates of the accuracy at 30, 50 and 70 trials per class. This time points were chosen because at 30 trials per class, the feature importance map showed first signs of a rating. 50 trials per class is used by Steyrl for his training period in his BCI system [29], and 70 trials per class represents the last training point which all naïve subjects had in common. The average MSE of the difference between the peak accuracy and the estimate is calculated in order to investigate the performance. Figure 3.18 shows the oob error rate for all naïve subjects over all training points. The red dotted line shows the mean value of the error rate.



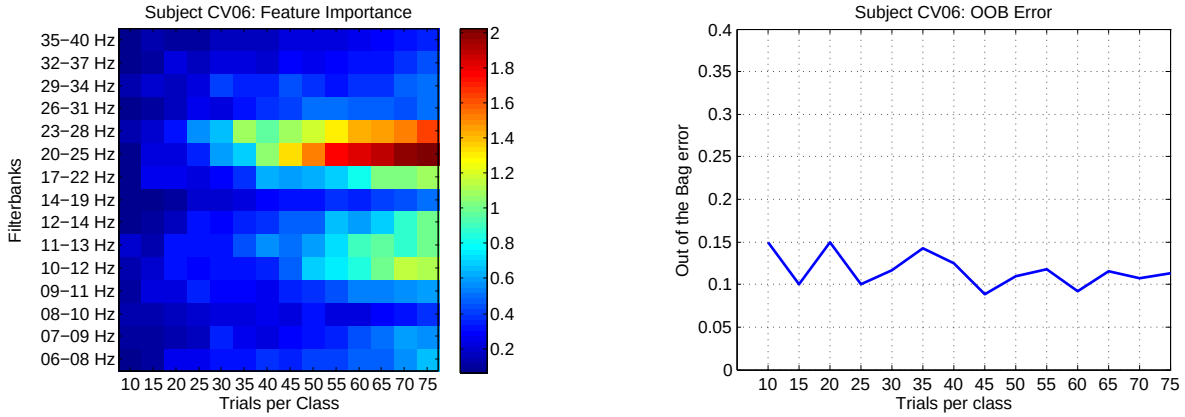
(a) Feature Importance (b) OOB-Estimate

Figure 3.9: CO07: Feature importance and OOB-Estimate



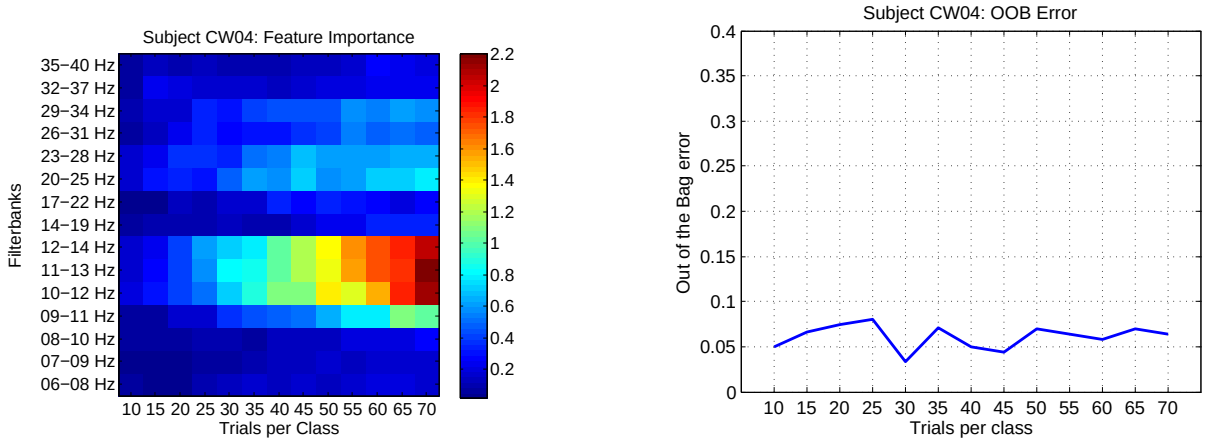
(a) Feature Importance (b) OOB-Estimate

Figure 3.10: CV04: Feature importance and OOB-Estimate



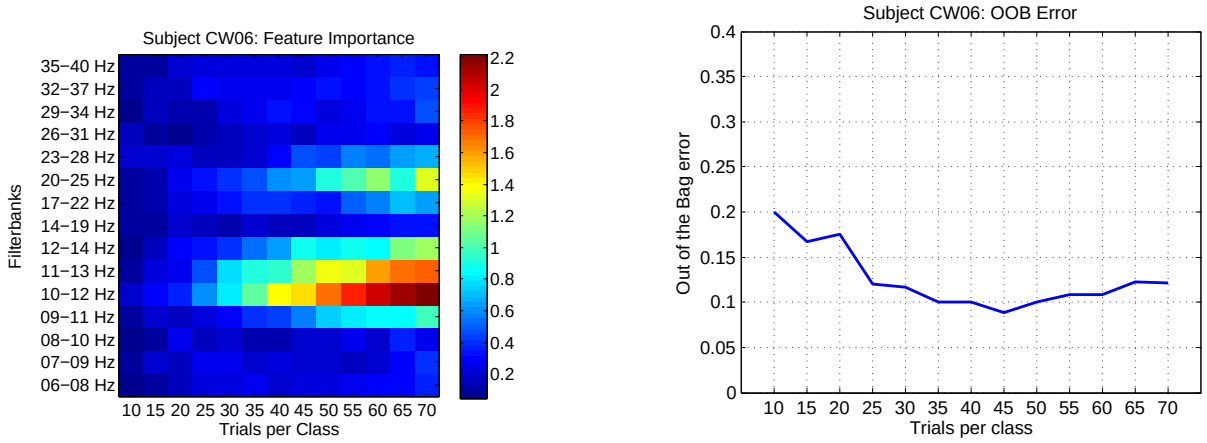
(a) Feature Importance (b) OOB-Estimate

Figure 3.11: CV06: Feature importance and OOB-Estimate



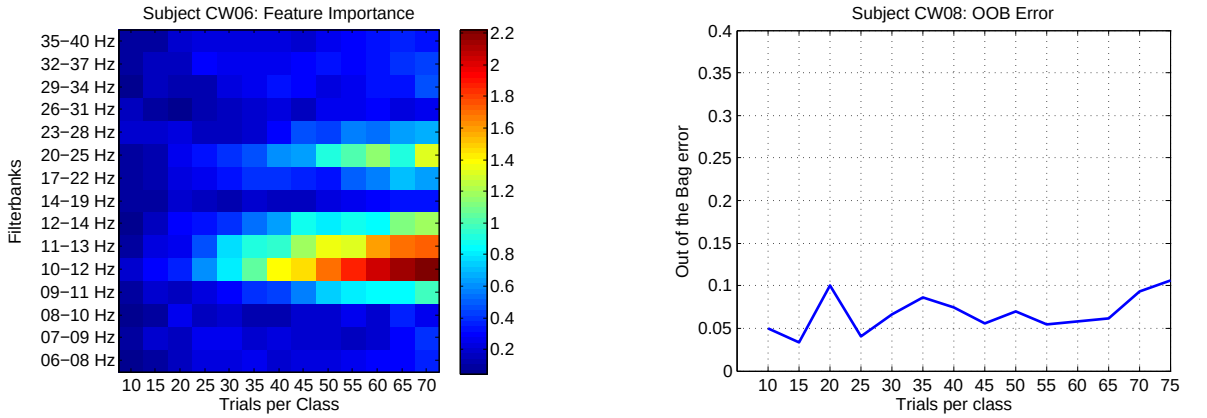
(a) Feature Importance (b) OOB-Estimate

Figure 3.12: CW04: Feature importance and OOB-Estimate



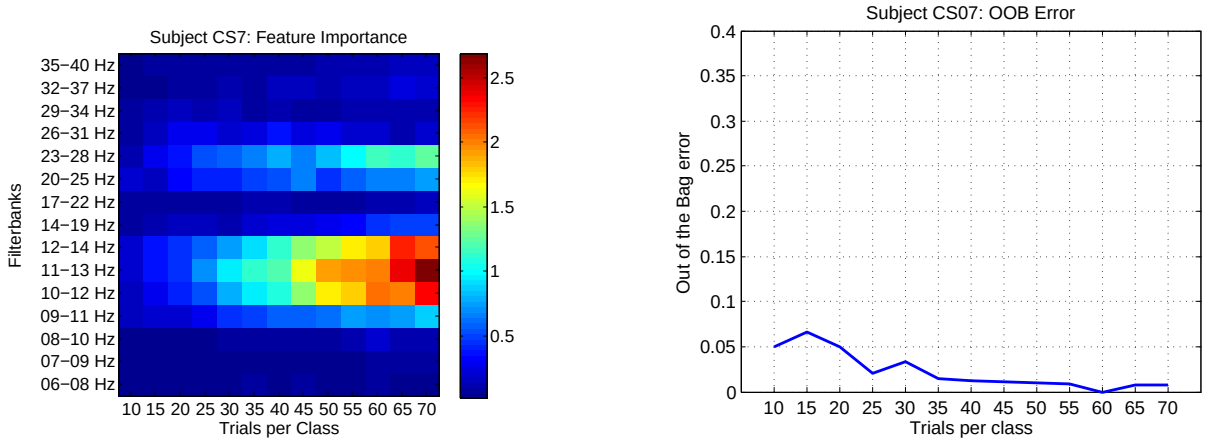
(a) Feature Importance (b) OOB-Estimate

Figure 3.13: CW06: Feature importance and OOB-Estimate



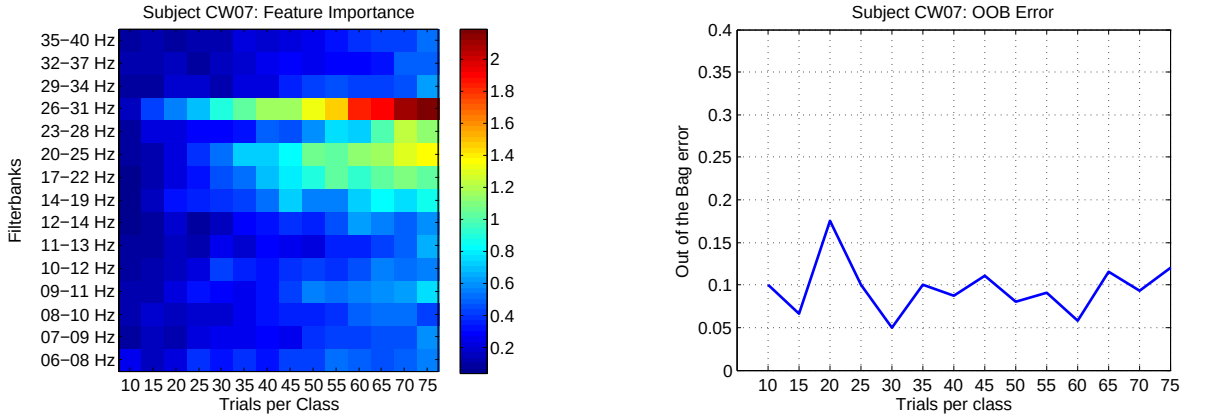
(a) Feature Importance (b) OOB-Estimate

Figure 3.14: CW08: Feature importance and OOB-Estimate



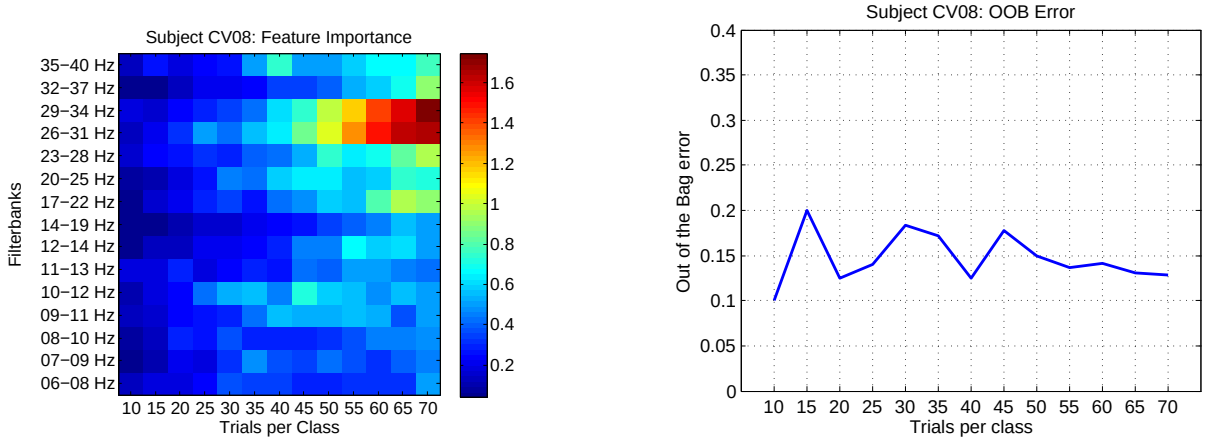
(a) Feature Importance (b) OOB-Estimate

Figure 3.15: CS07: Feature importance and OOB-Estimate



(a) Feature Importance (b) OOB-Estimate

Figure 3.16: CW07: Feature importance and OOB-Estimate



(a) Feature Importance (b) OOB-Estimate

Figure 3.17: CV08: Feature importance and OOB-Estimate

Comparison of peak accuracy and the estimated accuracy (oob)

True versus Predicted		OOB Accuracy Estimates					
subject	peak [%]	30 TPC [%]	Δ_{30}	50 TPC [%]	Δ_{50}	70 TPC [%]	Δ_{70}
CO07	86.43	88.33	1.90	86.00	0.43	90.71	4.29
CV04	61.43	78.33	16.90	74.00	12.57	73.57	12.14
CV07	89.29	91.67	2.38	94.12	4.83	89.58	0.30
CW06	90.71	88.33	2.38	90.00	0.71	87.86	2.86
CW04	88.57	96.67	8.10	93.00	4.43	93.57	5.00
CW08	86.43	93.33	6.90	93.00	6.57	90.71	4.29
CS07	98.57	96.67	1.90	99.00	0.43	99.29	0.71
CW07	79.29	95.00	15.71	92.00	12.71	90.71	11.43
CV08	82.86	81.67	1.19	85.00	2.14	87.14	4.29
Average MSE			**2.86**		**2.25**		**2.12**

Table 3.6: Comparison of peak accuracy and the estimated accuracy (oob): Δ describes the difference between the peak accuracy and the oob estimate at the designated training point. For each training point (30, 50 70 trials per class) the MSE of the difference between the peak accuracy and the estimate is calculated.

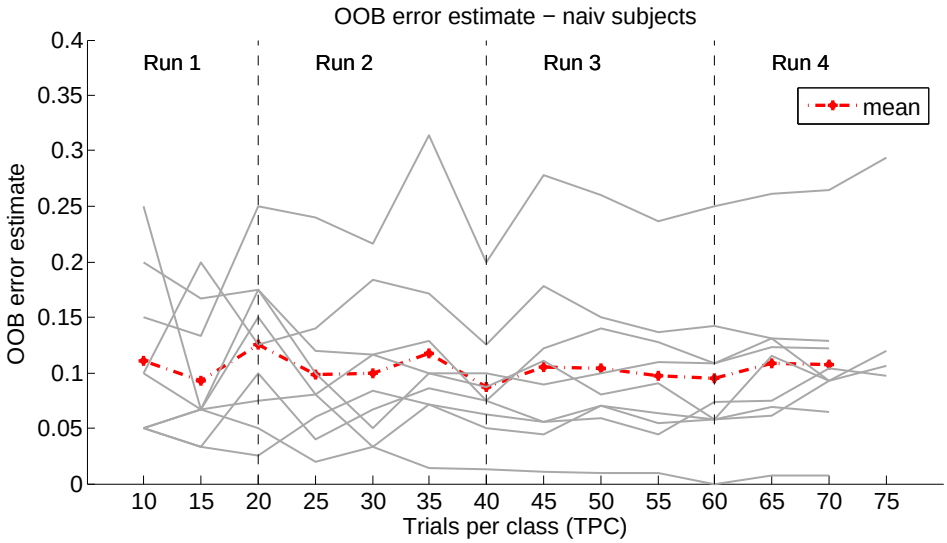


Figure 3.18: Out of the bag error: Plot of all naïve subjects. The red dotted line indicates the overall mean value

4 Discussion

4.1 Software

The idea of an independent Optimizer instance made it necessary to work with two different MATLAB instances. One part of the whole system has to run in real time $\pm 100ms$ for online data acquisition, paradigm and feedback for the subject, while the second part has to run asynchronously to the online part, since Outlier Rejection and retraining takes more time with the increase of the data available for retraining.
For communication between the systems, several methods were screened for the task:

- **Interprocess communication (IPC) via shared memory** seemed a proper way of communication but was forfeit since on the one hand creating semaphore/lock logic would cost a substantial amount of manpower and divert resources from the actual task at hand and on the other hand, the system would be limited to one single computer.
- **Data exchange via files** would have been a way to avoid dealing with semaphores and locks by creating and moving files. The idea was actually implemented in a system but lacked - as IPC before - the possibility to raise the system in a distributed way.
- **Communication via TCP/IP** using pnet [28] seemed the most legit way since the whole optimizer could be allocated on another PC which could work out beneficial for further more performance demanding improvements. The second aspect was that Faller et al. used the same approach for Communication [10], [41] which indicated that the approach seemed viable.

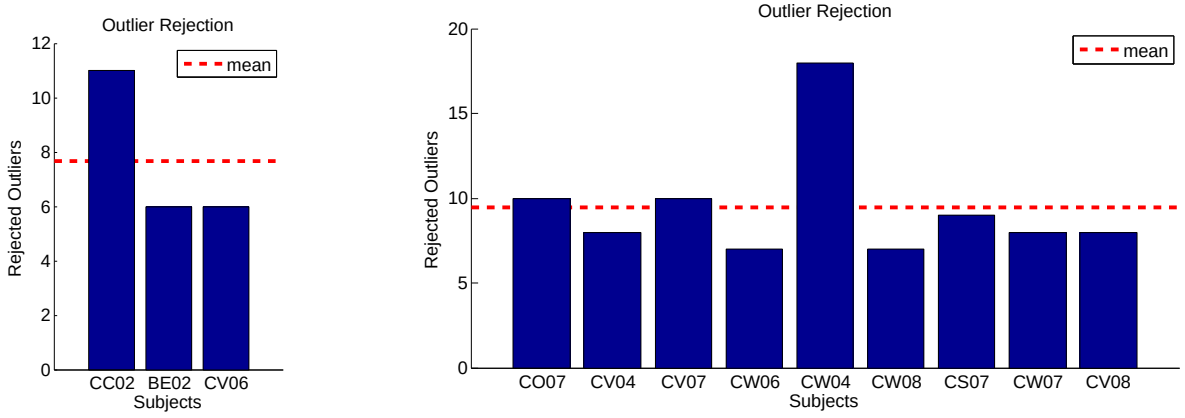
Contrary to Faller et al [10], the traffic between both instances is increased exponentially, since the trials sent to the Optimizer consist of not only 20, but 225 channels (208 for the filterbanks and the raw signal, rest for flags and events). Moreover the structure of the Random Forest classifier was far more complex than Fallers weighted matrix for the LDA classifier. Pnet is actually not able to send data in form of the MATLAB data-type *struct*, which is unfortunately the container format for the Random Forest classifier model. Therefore every new trained classifier is serialized using Fastserialze [31], before sent and deserialized at the Online System.
The system itself did undergo an extensive period of system tests using soft -and hardware signal generator to assure functionality. After overcoming all issues, the whole system worked reliable

for all subject-measurements.

4.1.1 Optimizer System

Outlier Rejection

The Outlier Rejection dropped in mean 9 of 160 trials for each subject. Detailed rates for each subject are shown in figure 4.1. Evaluating the performance of an artefact rejection method is quite a delicate task to do since the basis of clean EEG is ill-defined and mostly based on hypothesis, as it is for this system:
When looking at the feature importance plots (see figure 3.6 and ff.), it can be seen that the feature bands important to the Random Forest classifier are narrow banded in a few filterbanks and are not located in areas characteristic to artefacts. Eye movement is present in low frequency areas up to 9 Hz, while muscle movements are located in higher frequency areas around 30 to 40 Hz and have a more broad-banded characteristic. Since no features in the low frequency range of 6-9Hz are considered important and no widespread characteristic appear in the feature importance maps of subjects with performance rates at 80 plus percent, it can be **assumed** that the classifier was not trained by EOG or EMG.



(a) non-naïve (b) naïve

Figure 4.1: Outlier Rejection: Detailed drop rates for each subject

Nevertheless the statistical outlier rejection method is far from perfect, and based on the characteristics of the majority of the analysed data. If a user, for instance, blinks in every reference interval, or performs eye movement - these artefacts influence the the base line, leading to higher/odd standard derivations, may drive the classifier and mask other artefacts.
This assumption is not as odd as it may sound, since the paradigm is presented visually to the user. Investigations to resolve this issue are currently done.

4.1.2 Optimizer Performance

With increasing amount of training data available, the time for rejecting outliers and updating filters and classifier increases. For 150 trials, one complete update cycle on a second generation Intel Core i5 with 8 Gb RAM lasts about 14-15 seconds. With further increasing number of trials, the TCP/IP buffer runs the risk of overflowing, since the cycle time increases further and the need for temporal storing multiple data packages becomes imminent. To prevent this issue

in advance and in the aspect of future investigations, the Optimizer was - in contradiction to Faller et al. [10] - redesigned. Instead of handling just one new data package per update cycle, the Optimizer is now able to read the full TCP/IP buffer, allowing it to handle all available data packages per update cycle. This simple adaption enables the system the handling of indefinite vast amount of trials.

4.2 Hardware and Performance

Preliminary implementations were done on a first generation Intel Core i5 with 8 Gb RAM with Windows 7 as operating system. Both system instances were running on the same PC. After extensive functionality and performance tests, the system was migrated to the main PC of the laboratory, which has as basis a second Generation Intel Core i7 and 8 Gb RAM. Neither on the development nor on the laboratory system did any performance issues occur.
Moreover it is believed that a notebook with the same basis hardware as the development system is perfectly capable of providing sufficient performance for running the system, since a number of performance tests have been absolved on a similar notebook.
This feature is absolutely vital for extensive out of the lab field studies.

4.3 Data analysis

The split between naïve and non-naïve subjects has been done to exclude any kind of training effects resulting from previous studies. The main idea behind using non-naïve volunteers was to check whether or not the system is working in general and to uncover unknown issues.
The preliminary tests with the non-naïve volunteers showed not only extraordinary good results in peak accuracy (see table 3.2) but even high values for mean and median, which indicates to a long and stable classification period. In figure 3.2 the detection rates for each class are displayed. All three volunteers show a bias towards the feet class. Highest classification rates are achieved around 1.5 seconds after the onset of cue.
When looking at the feature importance maps of the three non-naïve volunteers (see figures 3.6(a),3.7(a),3.8(a)) the classifier chooses only a small number of features to be important for classification. Theses areas of importance are narrow-banded and gain more weight with every update. These areas lie in typical regions α (11-13 Hz) or medium (15-21 Hz) to high (21-38 Hz) β -band.

Since the results taken from taken non-naïve volunteers were promising, the next logical step was to test the system on volunteers who never operated a SMR based BCI before. All subjects performed significantly higher than chance (chance level = 59.6, $\alpha = 0.01$, see table 3.1) with a peak accuracy of 84.9 $\pm$ 10.3% (median 78.0%). Only two volunteers, CW6 and CV8, showed a bias towards any class (see figures 3.5(a), 3.5(e)). The feature importance maps looks similar to those of the non-naïve volunteers; narrow-banded importance areas in alpha and—or beta rythm regions.
Figure 4.2 shows a comparison between the actual designed BCI system (Schwarz2014), the BCI systems of Faller 2012 [10] and the Berlin BCI by Blankertz in 2008 [42]. The values for the boxplot of Faller2012 are taken from the first session. In direct comparison to Faller2012, Schwarz2014 achieves almost 10 percent higher peak accuracies over all subjects. Furthermore these accuracies are in a quite smaller range to find (compare the size of both boxes). Both systems operate with similar parameters: The calibration phase with 10 trials per class is the same, also the timepoints for recurrent updates. Faller recorded 100 trials per class, Schwarz2014

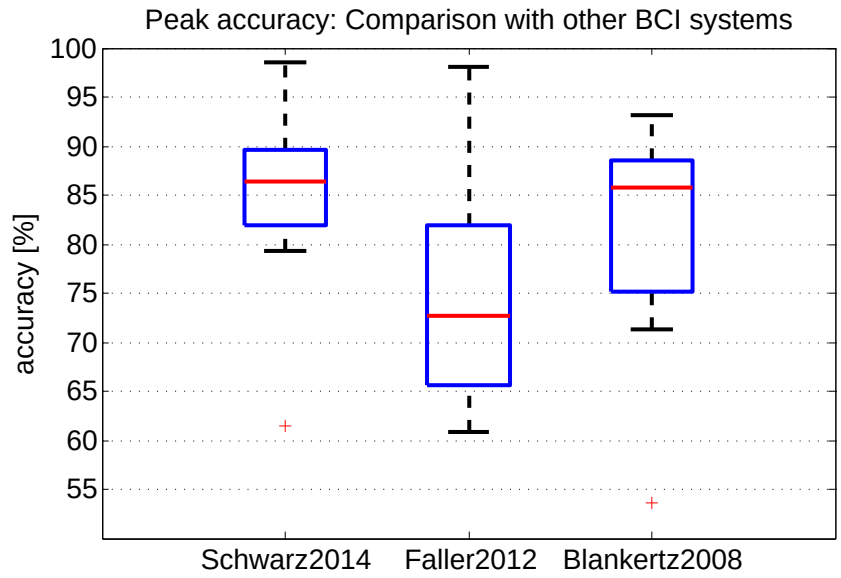


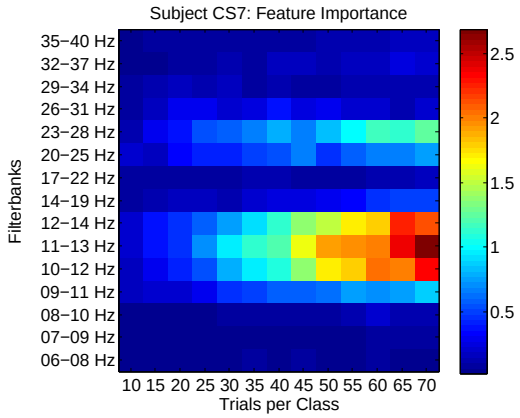
Figure 4.2: Comparison between the peak accuracies of the actual BCI system, Faller 2012 [10] and Blankertz 2008 [42]: The red line in the boxes represents the median of the accuracies.

recorded 80 trials per class. Blankertz2008 [42] represents a BCI system which is considered effective, well-performing and state of the art. Although a direct comparison to Schwarz2014 can not be done since both paradigms are quite different, both systems seem to operate at the same performance level, whereas Blankertz2008 used more than 4 times more electrodes (55 in total) and 140 trials per class for calibration to achieve these results. This fact is of particular interest for applying a BCI system outside the lab at "home", since complexity and duration of montage or maintenance decrease drastically.

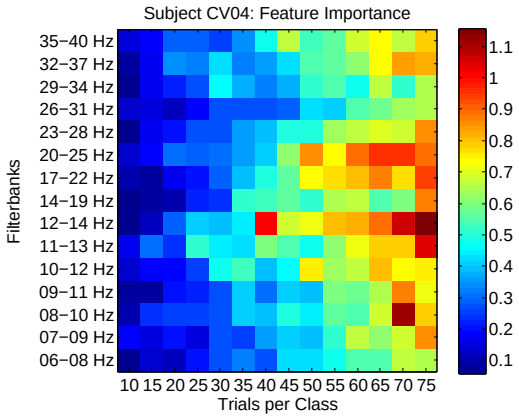
The colormaps (see 3.7(a) ff.) of the feature importance over time provide a quite useful tool for analysis and creating new assumptions. For all those subjects who performed in the area of 80 to 90 percent, the important features are located in narrow-banded areas. Furthermore, the importance of those bands begin to silhouette at around 25 to 30 trials per class against all other features.

While most of the volunteers performed in the range of 80 to 90 %, CV04 had a distinct worse performance of only 61.4 %, which lies only slightly above chance level. When looking at the feature importance map and compare it to the the best performer, as done in figure 4.3, well-marked differences can be observed. Compared to CS07, the important features are wide spread and parted over all filters, and the absolute values of the importance are not even half as high. If 4.3(b) seen individually, almost no statement or hypothesis can be given. The Outlier Rejection dropped almost twice as much trials than the usual average of nine trials. Since the runs were separated by short breaks, feedback from each user regarding personal comfort and opinions was gathered. Volunteer CV04 named problems to perform a motor imagery in each scheduled trial, and therefore left some aside. Furthermore CV04 deviated from the requested motor imagery for booth feet from run three on.

The "out of the bag" error appeared to be a quite accurate estimation of the peak accuracy, as can be seen in table 3.6. Three training points, 30,50 and 70 trials per class, were chosen for investigation. As can be seen, the average mean square error (MSE) results in under 3%. This allows an estimation towards the overall performance during the run, and may be applicable in



(a) Best naïve Performer: peak/mean/median: 0.986/0.922/0.936



(b) Worst naïve Performer: peak/mean/median: 0.614/0.538/0.536

Figure 4.3: Feature importance maps comparison between best (CV04) (see 4.3(a)) and worst performer (CS07) (see 4.3(b): Notice the narrow-banded property of the important features of subject CS07 while the important features of CV04 are widely spread around almost every filter. Moreover is the maximum mean drecrease in the Gini Index not even half the size in comparison to CS07.

remodelling the BCI to a performance driven system.
Figures 3.7(b)ff. and figure 3.18 show the projection of the OOB estimate over the updating points for each user. Most reached a local minimum at the beginning or during the third run, afterwards the OOB error seems to increase in subsequent updates again. This increase coincides with the the last run of the session. When questioning the designated volunteers, they stated increased fatigue and a decrease of concentration.

4.4 Classifier Analysis

The Random Forest classifier in its current implementation left a most formidable impression on the whole project. The provided intrinsic methods extend the implementation to be not only a classifier, but moreover to a data-analysis toolbox, although only a small number of methods are used. They concede to establish further and complex hypothesis, as can be seen in the feature importance plots constructed for the analysis in this particular system, where a large number of information (and assumptions) had been derivated from.
Since the algorithm is based on decision trees, it is able to discern between more than two classes, which could be quite beneficial in further advancements of the system.
The OOB estimates allow quite useful and accurate projections of the expected error rate and can be used to estimate time points, where the system works at optimum parameters and further adaptation can be renounced.
Another intrinsic feature of the Random Forest classifier is to weight classes individually. Although there was no use for this feature in this system, when introducing a third class, which might represent a resting state, it could become vital for practical use.

Overall, the algorithm seems a useful and resourceful candidate for the further development of future BCI systems.

4.5 Study shortcomings and possible future improvements

When dealing with EEG data as source, artefacts are always an issue who must be dealt with in order to evaluate the performance of BCI systems. The statistical outlier rejection implemented in this system removed contaminated trials, meaning that on average, at least 6 percent of all trials were dropped and are no further use to the investigator. Another approach to handle artefacts will be instead of dropping artefact contaminated trials, to remove the artefacts themselves. This would lead -in this case- to at least 6 percent more data for further investigations and moreover, lead to a far more stable system when trying to apply the system "out of the lab". There are already systems implemented which profess reliable artefact removal, as can be seen in [43] and [44], but most of those approaches lack of considerably in performance for real-time application (at least one second delay.)

Feedback is considered to play an important role in learning how to operate a BCI system. One of the main goals of this thesis was to provide feedback as soon as possible to the subject. Nevertheless the first 10 trials per class remain without feedback due to the fact that the CSP and the Random Forest do need a certain amount of basis data to perform appropriately. Sham-Feedback, as described in [45] and [46] could be used to encourage the subject even in its first trials. Of course the feedback must not be as well-marked as the real feedback after 10 trials per class, but even if the feedback bar grows only to e.g. 30 to 40 percent length, it would suffice and a seamless transition to real feedback could be established.

As already mentioned in the previous chapter, the length of the session was an issue to some of the subjects, which resulted in decreased performance in the last run. To counter this effects two possible approaches are viable for discussion:

a.) Shorten the whole session.

b.) Increase motivation and focus of the subject.

Shortening the session would be the easiest way to avoid this issue, but since fewer trials would be recorded, the study would drift into statistical insignificance and incomparability to other approaches. Moreover, in the long run, this kind of system should be designed to enable its user control over long periods of time. So the increase of focus and motivation of the subject seems the only legit way.

The solution for this issue may lie in the entertainment industry, or more specifically said, in modern computer games. With the rapid proceedings in technology, today's computer games build up entire worlds with foto-realistic graphical details to keep the attention of the user on the game. Storyboards and scenarios deepen the experience even further, while simultaneously the means of handling and interacting simplify to a point where only a few buttons are necessary. This trend of simplifying the handling evolved even further with the upcoming of mobile devices.

Computer games thrill and mesmerise the user throughout continuous hours, so it seems an obvious move to apply these systems as "paradigm", instead of the usual arrows and bars. There have been already made decisive steps towards this approach [8][9], but most state of the art systems do still rely on very simple and fatiguing paradigms.

The current system does perform recurrent updates in order to adapt to the new input trials. The OOB estimates showed a local minima in the third run of most of the subjects - afterwards it increased again. This might indicate a point where the adaption could end and the system might work on this level of "peak" performance. This would make it necessary to change the character of the system from a recurrent updating to a performance orientated system. The optimizer could still be in place and gather data, and calculate new classifier and filter modules, but should only update when e.g. the OOB error is significantly lower than the OOB of the

actually used one. Of course, further dependencies will have to be identified and dealt with.
The modulation of the system to deal with three or more classes would be another approach to improve functionality and increase the use of the system even further. The introduction of a "resting class" would pave the way from a synchronous Interface to an asynchronous approach, where the user is able to trigger actions not cue based, but at his discretion. In this connection, the intrinsic feature for weighting individual classes of the Random Forest could be of use: Since the resting class should be "active" most of the time, and commands are only short "disruptions" of this state, the resting class could get additional weighting, in cotrast to the other "command" classes.
So far, all experiments have been executed in a fully controlled, even shielded environment. It has not been evaluated whether or not the systems performs similar well in a non-laboratory environment. So, before starting on improving details of the system, it should be tested outside a controlled environment.

5 Conclusion

The aim of this work was to investigate the feasibility of the combination of state of the art technologies to form up an operational two class brain computer interface. The technologies used were the Filterbank CSP ([17],[22]) to ensure maximum class-separability for the logarithmic bandpower features and Random Forests for classification ([27], [13], [14]), as well as recurrent updates for classifier and filter([10], [41]).
The system is able to give feedback to the user within minutes and therefore actively supports training of right hand and feet motor imagery. To evaluate the performance, a supporting study using 9 novice users was done were all participants performed not only significantly above chance level, but acquired a peak accuracy of 84.9 $\pm$ 10.3% (median 78.0%).
In comparison, this 13 channel BCI surpasses Faller et al. BCI in peak accuracy by 10 % and performs well enough to be competitive with the 55 channel Berlin BCI by Blankertz [42]. The implementation of the system is considered a full success and awaits the next maturation level and improvements as described in section 4.5.
The ultimate goal should be an expedient solution to help users with severe motor impairment.

Bibliography

[1] J. J. Vidal, "Toward direct brain-computer communication," *Annual Review of Biophysics and Bioengineering*, vol. 2, pp. 157–180, 1973.

[2] B. Hamadicharef, "Brain-computer interface (bci) literature - a bibliometric study," in *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*. IEEE, 2010, pp. 626 – 629.

[3] S. G. Mason, A. Bashashati, M. Fatourechi, K. F. Navarro, and G. E. Birch, "A comprehensive survey of brain interface technology designs," *Annals of Biomedical Engineering*, vol. 35, pp. 137–169, 2007.

[4] M. Naeem, *Spatio-Temporal Decomposition of Bioelectrical Brain Signals*. Shaker Verlag, 2008.

[5] J. A. Kiernan. (2007) Anatomy 530a. lecture notes. The University of Western Ontario, Department of Anatomy and Cell Biology. [Online]. Available: http://instruct.uwo.ca/anatomy/530/530notes.htm

[6] G. Pfurtscheller and A. Aranibar, "Evaluation of event-related desynchronization (ERD) preceding and following voluntary self-paced movements," *Electroencephalography and Clinical Neurophysiology*, vol. 46, pp. 138–146, 1979.

[7] C. Brunner, M. Billinger, C. Vidaurre, and C. Neuper, "A comparison of univariate, vector, bilinear autoregressive, and band power features for brain - computer interfaces," *Medical & Biological Engineering & Computing*, vol. 49, no. 11, pp. 1337–1346, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11517-011-0828-x

[8] D. Marshall, D. Coyle, S. Wilson, and M. Callaghan, "Games, gameplay, and bci: The state of the art," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 5, no. 2, pp. 82–99, June 2013.

[9] C. Kapeller, C. Hintermüller, and C. Guger, "Augmented control of an avatar using an ssvep based bci," in *Proceedings of the 3rd Augmented Human International Conference*, ser. AH '12. New York, NY, USA: ACM, 2012, pp. 27:1–27:2. [Online]. Available: http://doi.acm.org/10.1145/2160125.2160152

[10] J. Faller, C. Vidaurre, T. Solis-Escalante, C. Neuper, and R. Scherer, "Autocalibration and recurrent adaptation: Towards a plug and play online ERD-BCI," *IEEE Transactions on Neural Systems Rehabilitation Engineering*, vol. 20, no. 3, pp. 313–319, May 2012.

[11] N. Brodu, F. Lotte, and A. Lecuyer, "Comparative study of band-power extraction techniques for motor imagery classification," in *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*, April 2011, pp. 1–6.

[12] D. J. McFarland, L. M. McCane, S. V. David, and J. R. Wolpaw, "Spatial filter selection for EEG-based communication," *Electroencephalography and Clinical Neurophysiology*, vol. 103, pp. 386–394, 1997.

[13] D. Steyrl, R. Scherer, G. R. Mueller-Putz, A. Holzinger(ed.), and G. Pasi(ed.), Eds., *Random Forests for Feature Selection in Non-invasive Brain-Computer Interfacing*. Springer, 2013.

[14] D. Steyrl, "On the suitability of random forests for detecting mental imagery for non-invasive brain-computer interfaces," 2012.

[15] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan, "Filter bank common spatial pattern (fbcsp) in brain-computer interface," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, June 2008, pp. 2390–2397.

[16] G. Dornhege, B. Blankertz, M. Krauledat, F. Losch, G. Curio, and K.-R. Mueller, "Combined optimization of spatial and temporal filters for improving brain-computer interfacing," *IEEE Transactions on Biomedical Engineering*, vol. 53, pp. 2274–2281, 2006.

[17] H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial EEG during imagined hand movement," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, pp. 441–446, 2000.

[18] Z. J. Koles, M. S. Lazar, and S. Z. Zhou, "Spatial patterns underlying population differences in the background EEG," *Brain Topography*, vol. 2, pp. 275–284, 1990.

[19] J. Mueller-Gerking, G. Pfurtscheller, and H. Flyvbjerg, "Designing optimal spatial filters for single-trial EEG classification in a movement task," *Clinical Neurophysiology*, vol. 110, pp. 787–798, 1999.

[20] R. TOMIOKA, G. DORNHEGE, G. NOLTE, B. BLANKERTZ, K. AIHARA, and K.-R. MUELLER, *Spectrally Weighted Common Spatial Pattern Algorithm for Single Trial EEG Classification*, ser. Mathematical engineering technical reports. Department of Mathematical Informatics, Graduate School of Information Science and Technology, the University of Tokyo, 2006. [Online]. Available: http://books.google.at/books?id=02S3XwAACAAJ

[21] W. Wu, X. Gao, B. Hong, and S. Gao, "Classifying single-trial eeg during motor imagery by iterative spatio-spectral patterns learning (isspl)," no. 55, pp. 1733–1743, June 2008.

[22] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Mueller, "Optimizing spatial filters for robust EEG single-trial analysis," *IEEE Signal Processing Magazine*, vol. 25, pp. 41–56, 2008.

[23] L. Breiman, *Classification and regression trees*, ser. The Wadsworth and Brooks-Cole statistics-probability series. Chapman & Hall, 1984. [Online]. Available: http://books.google.at/books?id=JwQx-WOmSyQC

[24] L. Breiman and L. Breiman, "Bagging predictors," in *Machine Learning*, 1996, pp. 123–140.

[25] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998. [Online]. Available: http://dx.doi.org/10.1109/34.709601

[26] IEEE, Ed., *An Empirical Comparison of Supervised Learning Algorithms*, 2006.

[27] J. F. Trevor Hastie, Robert Tibshirani, *The elements of statistical learning*, second edition ed. Springer, 2009.

[28] P. Rydesaeter. (2008) Tcp/udp/ip toolbox 2.0.6, pnet. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/345-tcpudpip-toolbox-2-0-6/content/tcp_udp_ip/pnet.m

[29] Mathworks. (2012) Matlab r2012a. [Online]. Available: http://www.mathworks.de

[30] A. L. Abhishek Jaiantilal. (2009) Random forest matlab implementation. [Online]. Available: https://code.google.com/p/randomforest-matlab/downloads/detail?name=RF_MexStandalone-v0.02.zip

[31] C. Kothe. (2012) Fast serialize/deserialize. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/34564-fast-serializedeserialize/content/hlp_serialize.m

[32] C. Breitwieser, C. Neuper, and G. Mueller-Putz, "Tia – standardizing raw biosignal delivery in bcis," in *5th International Brain-Computer Interface Conference 2011*, 2011, pp. 336–339. [Online]. Available: http://www.tobi-project.org/sites/default/files/public/Publications/TOBI-161.pdf

[33] J. del R. Millan. (2008-2013) The tobi project. [Online]. Available: http://www.tobi-project.org/

[34] G. R. Mueller-Putz, R. Scherer, D. Steyrl, and J. Faller. (2014) Graz-bci, libraries. [Online]. Available: http://bci.tugraz.at/

[35] M. Fatourechi, A. Bashashati, R. K. Ward, and G. E. Birch, "{EMG} and {EOG} artifacts in brain computer interface systems: A survey," *Clinical Neurophysiology*, vol. 118, no. 3, pp. 480 – 494, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1388245706015124

[36] M. S. Delorme A, "eeg-lab: an open source toolbox for analysis of single-trial eeg dynamics." pp. 9–21, 2004.

[37] T. S. Arnaud Delorme and S. Mageig, "Enhanced detection of artifacts in eeg data using higher-order statistics and independent component analysis," 2006.

[38] R. Scherer, "Towards practical brain-computer interfaces: Self-paced operation and reduction of the number of eeg sensors," 2008.

[39] gTec. (2014) g.gammasys - active electrode system. [Online]. Available: http://www.gtec.at/Products/Electrodes-and-Sensors/g.GAMMAsys-Specs-Features

[40] H. H. Jasper, "Report of the committee on methods of clinical examination in electroencephalography: 1957, the ten-twenty electrode system of the international federation," *Electroencephalography and Clinical Neurophysiology*, vol. 10, no. 2, pp. 370 – 375, 1958. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0013469458900531

[41] J. Faller, S. Torrellas, F. Miralles, C. Holzner, C. Kapeller, C. Guger, J. Bund, G. Mueller-Putz, and R. Scherer, "Prototype of an auto-calibrating, context-aware, hybrid brain-computer interface," in *34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012.

[42] B. Blankertz, F. Losch, M. Krauledat, G. Dornhege, G. Curio, and K.-R. Mueller, "The Berlin Brain-Computer Interface: accurate performance from first-session in BCI-naive subjects," *IEEE Transactions on Biomedical Engineering*, vol. 55, pp. 2452–2462, 2008.

[43] I. Daly, M. Billinger, R. Scherer, and G. Mueller-Putz, "On the automated removal of artifacts related to head movement from the EEG," *IEEE Transactions on neural systems and rehabilitation engineering*, 2013.

[44] I. Daly, N. Nicolaou, S. Nasuto, and K. Warwick, "Automated artifact removal from the electroencephalogram; a comparative study," *Clinical EEG and neuroscience*, vol. (onlien fi, 2013.

[45] A. Barbero Jimenez and M. Grosse-Wentrup, "Biased feedback in brain-computer interfaces," *Journal of NeuroEngineering and Rehabilitation*, vol. 7, no. 34, pp. 1–4, 7 2010.

[46] F. Lotte, F. Larrue, and C. Mühl, "Flaws in current human training protocols for spontaneous brain-computer interfaces: lessons learned from instructional design," *Frontiers in Human Neuroscience*, vol. 7, no. 568, 2013. [Online]. Available: http://www.frontiersin.org/human_neuroscience/10.3389/fnhum.2013.00568/abstract

A Appendix

A.1 Online System - MATLAB/Simulink Implementation Details

The following section depicts the implementation of the Online Sytem and the critical parts of its implementation. Notice that the Online System is implemented using Simulink. Any provided code is integrated as S-functions to the model.

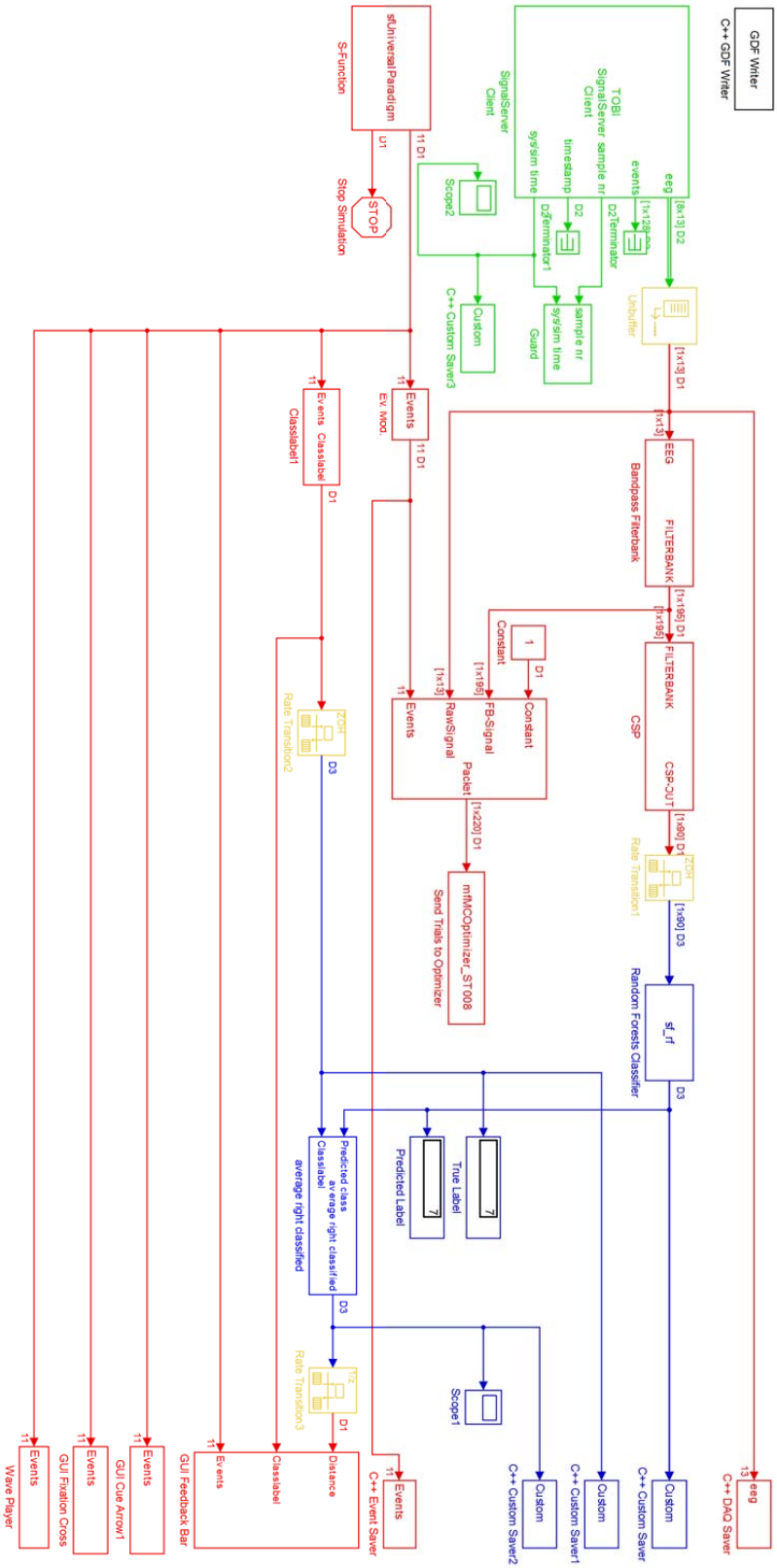


Figure A.1: The Simulink representation of the Online Model. Note that there are section with different colour representiations. These are indicators for different clock rates which are achieved by installed rate transitions.

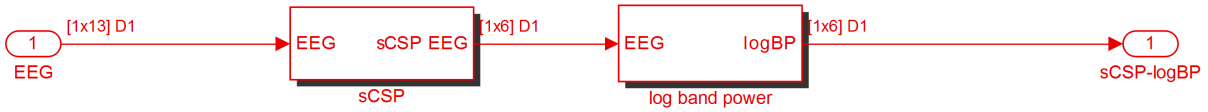


Figure A.2: Each output of the filterbank has its own csp filter and feature generator.

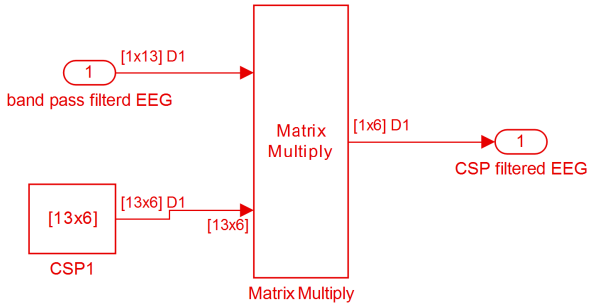


Figure A.3: The CSP block contains a simple matrix multiplication. The CSP filter itself is calculated at the Optimizer Instance.

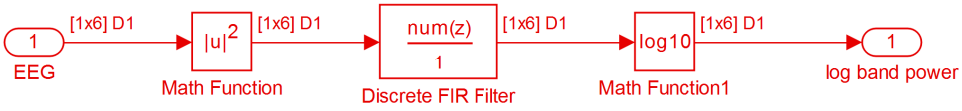


Figure A.4: The generation of the logarithmic bandpower features.

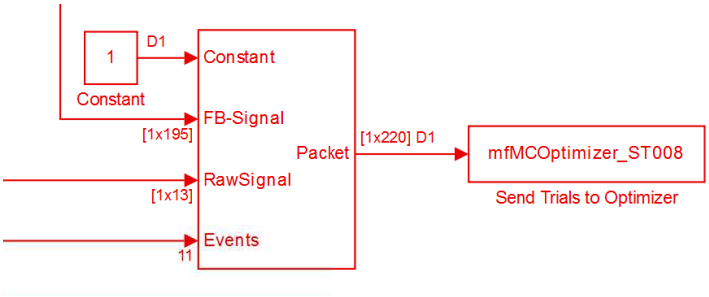


Figure A.5: The Mux merges the incoming signals and provides it to the Com-interface, which is realized as S-function; Notice the constat, whicj acts as a placeholder for the flag selector.

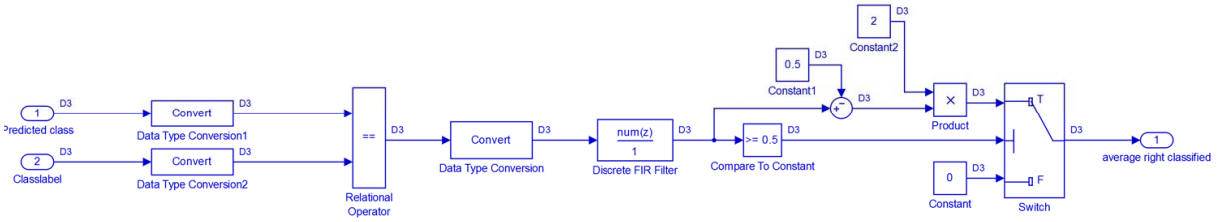


Figure A.6: Feedback logic for calculating the "strength" meaning the length/sizing of the feedback bar. (see figure 2.15

A.1.1 Code Listings for the Online System

COM-Interface: S-function

```
function [sys, x0, str, ts] = mfMCOptimizer_ST008 ( t, x, u, flag )
%
% mfSendTrials - Send acquired trials to external optimization algorithm
  global rtBCI;
  switch ( flag )
    %--------------------
    % Initialization
    %--------------------
    case 0
      rtBCI.mfSendTrials = [];
      rtBCI.mfSendTrials.AllowMultiSessionTraining = 1;
      rtBCI.mfSendTrials.SendEnabled  = 1;
      rtBCI.mfSendTrials.DestHostIP   = '127.0.0.1';
      rtBCI.mfSendTrials.DestHostPort = 12360;
      rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS = 208; %RAW Channels
      rtBCI.mfSendTrials.NUM_TRANSMITTED_EVENTS  = 11; % # Events  per Sample
      rtBCI.mfSendTrials.NUM_SIGNALS_META        =  2;
      rtBCI.mfSendTrials.NUM_TOTAL_SIGNALS       = rtBCI.mfSendTrials.NUM_SIGNALS_META +
             rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS + rtBCI.mfSendTrials.NUM_TRANSMITTED_EVENTS;
      rtBCI.mfSendTrials.NUM_TOTAL_EVENTS        = rtBCI.mfSendTrials.NUM_SIGNALS_META +
             rtBCI.mfSendTrials.NUM_TRANSMITTED_EVENTS;
      rtBCI.mfSendTrials.UpdateEnabled = 1;
%
      if ( strcmpi ( rtBCI.OPERATION_MODE, 'STD_FB' ) )
        rtBCI.mfSendTrials.UpdateEnabled = 0;
      elseif ( strcmpi ( rtBCI.OPERATION_MODE, 'ADAP_ONL_TRN' ) )
        rtBCI.mfSendTrials.UpdateEnabled = 1;
      else
        rtBCI.mfSendTrials.UpdateEnabled = 1;
      end

      rtBCI.mfSendTrials.iCounter  = 0;
      rtBCI.mfSendTrials.iRecvTime = round ( rtBCI.Model.SampleRate / 8 );
      rtBCI.mfSendTrials.EV_TRIAL_START = 768; % EventCode
      rtBCI.mfSendTrials.TRIAL_LENGTH   = 9; % sec
      rtBCI.mfSendTrials.bInTrial = false;
      rtBCI.mfSendTrials.bStartupEvent = false;
      rtBCI.mfSendTrials.iCurTrialIdx = 0;
      rtBCI.mfSendTrials.aTrialPacket = zeros ( rtBCI.mfSendTrials.NUM_TOTAL_SIGNALS,
             rtBCI.mfSendTrials.TRIAL_LENGTH*rtBCI.Model.SampleRate );
      rtBCI.mfSendTrials.aEventPacket = zeros ( rtBCI.mfSendTrials.NUM_TOTAL_EVENTS  , 1 );
      rtBCI.mfSendTrials.iCurCFRIdx   = 0;
      rtBCI.mfSendTrials.iPacketNumber = 1;
      rtBCI.counter = 1;


      % Open TCP connection
      if ( rtBCI.mfSendTrials.SendEnabled )
        % Open connection
        rtBCI.mfSendTrials.tcpCon = pnet ( 'tcpconnect', rtBCI.mfSendTrials.DestHostIP,
               rtBCI.mfSendTrials.DestHostPort );
        % SET MAXIMUM WRITE-TIMEOUT / Block Simulink for this maximum time!
        pnet ( rtBCI.mfSendTrials.tcpCon, 'setwritetimeout', 1.25 );
        % SET MAXIMUM READ-TIMEOUT / Block Simulink for this maximum time!
        pnet ( rtBCI.mfSendTrials.tcpCon, 'setreadtimeout', 1.25);
        if ( rtBCI.mfSendTrials.tcpCon ~= -1 )
          display ( ['## [' GetTimeStamp() '] SendTrials, opened connection ['
                 num2str(rtBCI.mfSendTrials.tcpCon) '] to [' rtBCI.mfSendTrials.DestHostIP ':'
                 num2str(rtBCI.mfSendTrials.DestHostPort) ']'] );
          fprintf ( '## Read remaining things on startup [%s]\n\n', pnet ( rtBCI.mfSendTrials.tcpCon,
                 'read', [], [], 'network', [], 'noblock' ) );
        else
          display ( ['## [' GetTimeStamp() '] SendTrials, FAILED to open connection to ['
                 rtBCI.mfSendTrials.DestHostIP ':' num2str(rtBCI.mfSendTrials.DestHostPort) ']'] );
          rtBCI.mfSendTrials.SendEnabled = 0;
        end
      end

      [sys, x0, str, ts] = mdlInitializeSizes ( );
    %--------------------
    % Block output
    %--------------------
    case 3
      rtBCI.mfSendTrials.iCounter = rtBCI.mfSendTrials.iCounter + 1;
      if ( rtBCI.mfSendTrials.SendEnabled )
        % Check whether update is enabled and receive update time has
        % passed
        if ( rtBCI.mfSendTrials.UpdateEnabled && ( mod ( rtBCI.mfSendTrials.iCounter,
               rtBCI.mfSendTrials.iRecvTime ) == 0 ) )
          rtBCI.mfSendTrials.iCounter = 0;
          dRecvBuffer = [];
          dDataBuffer = [];
          tic;
          dRecvBuffer = pnet ( rtBCI.mfSendTrials.tcpCon, 'read', 1, 'double', 'network', [], 'noblock' );
          if ( ~ isempty ( dRecvBuffer ) )
            switch ( dRecvBuffer ( 1 ) )
              %% Client-Side Packet-CODE 50, receive classifier and parameter update!
              case 50
                 dDataBuffer = pnet ( rtBCI.mfSendTrials.tcpCon, 'read', 1, 'double', 'network', [],
                        'noblock' );
                if ( ~ isempty ( dDataBuffer ) )
```

```
82                   iLen = fix ( dDataBuffer ( 1 ) );
83  tic
84                   aDataBuffer = pnet ( rtBCI.mfSendTrials.tcpCon, 'read', iLen, 'uint8','network');
85  toc
86                   rtBCI.times(rtBCI.counter)  = toc;
87                   store =aDataBuffer';
88                   rtBCI.raw_store(rtBCI.counter,:,:) = length(store);
89                   store_it = hlp_deserialize ( store );
90                   rtBCI.modelRF.modelRF = [];
91                   rtBCI.modelRF.modelRF = store_it;
92                   rtBCI.sCSP.csp_filter = [];
93                   rtBCI.sCSP.csp_filter = rtBCI.modelRF.modelRF.CSP_matrices;
94                   for CSP_idx = 1:rtBCI.bandpass.NUM_FILTERBANK
95                   set_param ( ['fallout_model/CSP/sCSP_BP' num2str(CSP_idx) '/sCSP/CSP' num2str(CSP_idx)] ,
                            'Value', mat2str(squeeze(rtBCI.modelRF.modelRF.CSP_matrices(:,:,CSP_idx) )));
96                   end
97                   fprintf ( '## package arrived. [%s] TCP-FLAG [%d], [%d] \n', GetTimeStamp ( ),
                            dRecvBuffer(1), length(store));
98                   sTimestamp = strrep(num2str(fix(clock),'%02d'),' ','');
99                   if rtBCI.modelRF.modelRF.SaveMe == 1
100                  save([rtBCI.Subject.sCFRPath '\' 'CFR_Run' num2str(rtBCI.Subject.iRunNumber) '_'
                            num2str(rtBCI.counter) '_' num2str(rtBCI.Subject.ID) '_' sTimestamp ],
                            'store_it');
101                  rtBCI.storeroom(rtBCI.counter,:,:) = store_it;
102                  rtBCI.counter = rtBCI.counter +1;
103                  end
104                 end
105               otherwise
106                 fprintf ( '\n## [%s] TCP-FLAG [%d] unknown command.\n', GetTimeStamp ( ), dRecvBuffer(1) );
107             end
108           end
109         end
110         % Do events include startup event? Check if the actual package
111         % contains the startevent: look from u(14 to 25) for startevent. if
112         % yes, sum is greater than 1
113         if ( sum ( u(1+rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS+1:1+...
114                rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS+...
115                rtBCI.mfSendTrials.NUM_TRANSMITTED_EVENTS) == rtBCI.mfSendTrials.EV_TRIAL_START ) > 0 )
116           if ( ~ rtBCI.mfSendTrials.bStartupEvent )
117               display ( ['## in the loop'] );
118             rtBCI.mfSendTrials.bInTrial = true;
119             rtBCI.mfSendTrials.bStartupEvent = true;
120             rtBCI.mfSendTrials.iTrialTimeout = 0;
121           end
122         else
123
124         end
125
126         if ( rtBCI.mfSendTrials.bInTrial && ( rtBCI.mfSendTrials.iTrialTimeout < ( rtBCI.Model.SampleRate
                  * rtBCI.mfSendTrials.TRIAL_LENGTH ) ) )
127           rtBCI.mfSendTrials.iCurTrialIdx = rtBCI.mfSendTrials.iCurTrialIdx + 1;
128           rtBCI.mfSendTrials.aTrialPacket(2:end,rtBCI.mfSendTrials.iCurTrialIdx) =
                    u(1:rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS+...
129           rtBCI.mfSendTrials.NUM_TRANSMITTED_EVENTS+1)';
130           rtBCI.mfSendTrials.iTrialTimeout = rtBCI.mfSendTrials.iTrialTimeout + 1;
131         else
132           % If packet is full send it
133           if ( rtBCI.mfSendTrials.bInTrial && ( rtBCI.mfSendTrials.iCurTrialIdx == (
                    rtBCI.Model.SampleRate * rtBCI.mfSendTrials.TRIAL_LENGTH ) ) )
134             aClassOccs = [];
135             bCorrectCue = false;
136             for ( iCIdx = 1:length(rtBCI.mfSingleCLEventModifier.ALL_CLASSLABELS) )
137               aClassOccs(iCIdx) = sum ( sum (
                          rtBCI.mfSendTrials.aTrialPacket(rtBCI.mfSendTrials.NUM_SIGNALS_META+...
138                       rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS+1:end,:)...
139                       == rtBCI.mfSingleCLEventModifier.ALL_CLASSLABELS(iCIdx) ) );
140             end
141
142             if ( sum ( aClassOccs >= rtBCI.Model.SampleRate ) == 1 )
143               bCorrectCue = true;
144               iCue(1,1) = rtBCI.mfSingleCLEventModifier.ALL_CLASSLABELS(find(aClassOccs >=
                          rtBCI.Model.SampleRate ));
145             end
146             if ( bCorrectCue )
147               % Add line with true-label to the packet
148               rtBCI.mfSendTrials.aTrialPacket(1,:) = iCue;
149 %               rtBCI.mfSendTrials.aEventPacket = [iCue rtBCI.mfSendTrials.aEventPacket'];
150               % Add line with true-label to the packet
151               rtBCI.mfSendTrials.aTrialPacket(1,:) = iCue;
152               % Send packet
153               tic;
154               % Send Server-Side Packet-CODE 2, indicates incoming trial data packet
155               pnet ( rtBCI.mfSendTrials.tcpCon, 'write', 2 );
156               % Send number of sensors and number of events
157               pnet ( rtBCI.mfSendTrials.tcpCon, 'write', [rtBCI.mfSendTrials.NUM_TRANSMITTED_SENSORS
                          rtBCI.mfSendTrials.NUM_TRANSMITTED_EVENTS] );
158               dTime01 = toc;
159               tic;
160               % Send actual data packet as matrix of true classlabel info, time-idx, channels and events
161               pnet ( rtBCI.mfSendTrials.tcpCon, 'write', rtBCI.mfSendTrials.aTrialPacket );
162               dTime02 = toc;
163               fprintf ( '## [%s] Sent data packet [%d] true-label [%d], size [%s] took [%1.3f/%1.3fs]\n',
                          GetTimeStamp ( ), rtBCI.mfSendTrials.iPacketNumber, iCue(1,1),
                          num2str(size(rtBCI.mfSendTrials.aTrialPacket)), dTime01, dTime02 );
164               rtBCI.mfSendTrials.iPacketNumber = rtBCI.mfSendTrials.iPacketNumber + 1;
165             else
166               display ( ['## Could not find correct cue!'] );
167             end
```

```
168             else
169               if ( rtBCI.mfSendTrials.bInTrial )
170                 display ( ['## Inappropriate packet size [' num2str(rtBCI.mfSendTrials.iCurTrialIdx) ']'] );
171               end
172             end
173
174             rtBCI.mfSendTrials.bInTrial      = false;
175             rtBCI.mfSendTrials.iTrialTimeout = 0;
176             rtBCI.mfSendTrials.bStartupEvent = false;
177             rtBCI.mfSendTrials.iCurTrialIdx  = 0;
178           end
179         end
180
181         sys = [];
182       %--------------------
183       % Destructor
184       %--------------------
185       case 9
186         % Close TCP connection
187         if ( rtBCI.mfSendTrials.SendEnabled )
188           if ( rtBCI.mfSendTrials.AllowMultiSessionTraining )
189             % Only disconnect for next session
190             pnet ( rtBCI.mfSendTrials.tcpCon, 'write', 1 );
191           else
192             % Disconnect and close server
193             pnet ( rtBCI.mfSendTrials.tcpCon, 'write', 0 );
194           end
195           % Close connection
196           pnet ( rtBCI.mfSendTrials.tcpCon, 'close' );
197
198           display ( ['## [' GetTimeStamp() '] Disconnected from [' rtBCI.mfSendTrials.DestHostIP ':'
                  num2str(rtBCI.mfSendTrials.DestHostPort) ']'] );
199         end
200         % Remove all variables from rtBCI
201         rtBCI = rmfield ( rtBCI, 'mfSendTrials' );
202
203       %--------------------
204       % Unhandled flags
205       %--------------------
206       case { 1, 2, 4 }
207         sys = [];
208       %--------------------
209       % Unexpected flags
210       %--------------------
211       otherwise
212
213         error ( ['## Unhandled flag [' num2str(flag) ']'] );
214
215     end
216
217   end
218
219   %% Block initialization function
220   function [sys, x0, str, ts] = mdlInitializeSizes ( )
221   %
222   % mdlInitializeSizes
223   % Return the sizes, initial conditions, and sample times for the S-function.
224   %
225     sizes = simsizes;
226     sizes.NumContStates  =  0;
227     sizes.NumDiscStates  =  0;
228     sizes.NumOutputs     =  0;  % dynamically sized
229     sizes.NumInputs      = -1;  % dynamically sized
230     sizes.DirFeedthrough =  1;  % has direct feedthrough
231     sizes.NumSampleTimes =  1;
232
233     sys = simsizes(sizes);
234     str = [];
235     x0  = [];
236     ts  = [-1 0];   % inherited sample time
237   end
```

Listing A.1: COM-Interface - S-function

A.2 Optimizer System - MATLAB Implementation Details

The implementation of the Optimizer System is purely done in MATLAB, no additional simulink model is needed.

```
1   function [] = CSP_RF_Optimizer_ST008 ( sSubject, sSPATIAL_FILTER )
2   %% INPUT PARAMETER CHECK
3     if ( ~ exist ( 'sSubject', 'var' ) || ( nargin < 1 ) )
4       help MCOptimizer_ST008
5       error ( '## Please provide all necessary input parameters.' )
6     end
7     close all; pnet ( 'closeall' );
8     if ( ~ exist ( 'sStartFromFile', 'var' ) )
9
```

```
10
11      % CONFIGURATION AND INI SETTINGS
12      SAMPLE_RATE = 256;
13      DEBUG_OUTLIER_REJECTION = false;
14      NUMBER_OF_CLASSES           = 2;
15      ALL_CLASSLABELS             = [ 7 8];
16      ALL_CL_COMBINATIONS         = [ 7 8];
17      NUM_SIGNALS_META            = 2;
18      NUM_TRANSMITTED_SENSORS     = 208;
19      NUM_TRANSMITTED_EVENTS      = nan;
20      NUM_SIGNALS_TOTAL           = nan;
21      NUM_RAW_CHANNELS            = 13;
22      NUM_CHANNELS                = 13;
23      NUM_FILTERBANKS             =15;
24      pnet_labels                 = [];
25      stData.X_trn_storeroom              = [];
26      stData.aCurCLLabels_storeroom = [];
27      X_trn = [];
28      X_trn_counter = 1;
29      bins = 2:41;
30      ntrees = 1000;
31      mtry = 0;
32      extra_options.importance=1;
33      counter= 1;
34      TRIAL_STEP_SIZE             = 4;
35      MINIMUM_TRIALS_PER_CLASS    = 10;
36      LAST_MIN_CLASSES            = -15;
37      TRIAL_LENGTH                = 9;
38      CLEAR_BETWEEN_SESSIONS      = 0;
39
40      START_SEG_SAMPLE  = 4 * SAMPLE_RATE;
41      END_SEG_SAMPLE    = 8 * SAMPLE_RATE;
42      START_REF_SAMPLE  = fix ( 1 * SAMPLE_RATE );
43      END_REF_SAMPLE    = fix ( 2 * SAMPLE_RATE );
44
45      TRAINING_WINDOW_SIZE        = -1;    % 40
46
47      sLocalIP   = '127.0.0.1';
48      iLocalPort = 12360;
49      sTimestamp = strrep(num2str(fix(clock),'%02d'),' ','');
50      sResultPath = ['./rec/RF_Optimizer/' sSubject '_' GetTimeStamp(3) '/' sSubject '_' sTimestamp '/'];
51      sResultfile = [sResultPath sSubject '_' sTimestamp '.mat'];
52      sDiary      = ['./rec/RF_Optimizer/' sSubject '_' GetTimeStamp(3) '/' sSubject '_' sTimestamp
              '/Diary_' sTimestamp '.txt'];
53
54      [sTP, sTN, sTE] = fileparts ( sResultfile );
55
56      CreateFolder ( sTP );
57      bFirst = true;
58
59      % Clear client session specific data.
60      iLastMinClasses = LAST_MIN_CLASSES;
61      iOutRejOffset = 0;
62
63      stData.aSignal       = [];
64      stData.aTriggers     = [];
65      stData.aTrueLabels   = [];
66      stData.bFirstAttempt = 1;
67
68      cOut = BCIClassifier ( );
69      aAllSigOutIdx = [];
70      INI = [];
71      w = [];
72
73      iPacketNumber = 1;
74      iCurrCFRIdx = 0;
75      iCurMinClasses = 0;
76      iOptIdx = 1;
77
78      %% General configuration
79      cOut = set ( cOut, 'sMethod',         'ERD' );
80      cOut = set ( cOut, 'iNSegments',      8 );
81      cOut = set ( cOut, 'iValsPerSegment', 4 );
82      cOut = set ( cOut, 'dAveraging',      1 );
83      cOut = set ( cOut, 'dTrialLength',    7 );
84      cOut = set ( cOut, 'dCueTime',        2 );
85      cOut = set ( cOut, 'bShowFigures',    false );
86      cOut = set ( cOut, 'FILTER_ORDER',    5 );
87      cOut = set ( cOut, 'sResPath',        sResultPath );
88
89      stHeader.SampleRate = SAMPLE_RATE;
90      % Set data manually
91      cOut = set ( cOut, 'stHeader', stHeader );
92
93    else
94      fprintf ( '\n## Starting Feature Optimizer from file [%s]\n', sStartFromFile );
95
96      load ( sStartFromFile );
97
98      whos
99    end
100
101 %% ENABLE DIARY
102   diary ( sDiary ); diary on;
103   fprintf ( '\n' );
104
105 %% OPEN SOCKET
106
```

```
107   iSock = pnet ( 'tcpsocket', iLocalPort );
108   % Socket opened successfully
109
110
111   if ( iSock ~= -1 )
112     fprintf ( '## [%s] Successfully started online feature optimizer @ %s:%d\n', GetTimeStamp(), sLocalIP,
              iLocalPort );
113     iCon = pnet ( iSock, 'tcplisten' );
114     bServerRunning = true;
115     % IF TRUE, CLEAR ALL THE DATA SAVED FROM PREVIOUS SESSION
116     if ( CLEAR_BETWEEN_SESSIONS )
117       iLastMinClasses = LAST_MIN_CLASSES;
118
119       stData.aSignal       = [];
120       stData.aTriggers     = [];
121       stData.aTrueLabels   = [];
122       stData.aFeatures     = [];
123       stData.bFirstAttempt = 1;
124
125       aAllSigOutIdx = [];
126       INI = [];
127       w = [];
128       iPacketNumber = 1;
129     end
130
131     % OPENING UP THE SERVER CONNECTION
132     while ( bServerRunning && ( iCon ~= -1 ) )
133       [aDestIP, iDestPort] = pnet ( iCon, 'gethost' );
134       fprintf ( '## [%s] NEW client connection @ %d/%d.%d.%d.%d:%d\n\n', GetTimeStamp(), iCon, aDestIP(1),
                aDestIP(2), aDestIP(3), aDestIP(4), iDestPort );
135
136       bClientConnected = true;
137
138            % Send initial classifiers if available
139       if ( isfield ( stData, 'modelRF' ) && ( ~isempty ( stData.modelRF )  ) )
140
141        sendtime01 = tic;
142        stData.modelRF.SaveMe = 0;
143        stData.modelRF_serialized  = hlp_serialize(stData.modelRF);
144        psize = size(stData.modelRF_serialized);
145        pnet ( iCon, 'write', 50 );
146        pnet ( iCon, 'write', size ( [stData.modelRF_serialized], 1 ) );
147        pnet ( iCon, 'write', [stData.modelRF_serialized],'uint8' );   % Send Indices and other Information
148        sendtime02= toc(sendtime01);
149        fprintf('     Initial Classifier sending DONE. Took [%3f] sec \n', sendtime02);
150       end
151
152       % Client connected
153       while ( bClientConnected )
154        wholeone = tic;
155
156        bMorePackets = true;
157        while ( bMorePackets && ( iCon ~= -1 ) && bClientConnected)
158         aBuffer = pnet ( iCon, 'read', 1, 'double', 'network' );
159         if ( ~isempty ( aBuffer ) )
160          switch ( aBuffer ( 1 ) )
161             %% Server-Side Packet-CODE 0, disconnect client connection and close optimization server
162             case 0
163               fprintf ( '\n## [%s] TCP-FLAG [%d], closing connection, quitting server\n', GetTimeStamp (
                        ), aBuffer(1) );
164               pnet ( iCon, 'close' );
165               pnet ( iSock, 'close' );
166               bClientConnected = false;
167               bServerRunning   = false;
168
169             %% Server-Side Packet-CODE 1, disconnect client connection
170             case 1
171               fprintf ( '\n## [%s] TCP-FLAG [%d], closing connection\n', GetTimeStamp ( ), aBuffer(1) );
172               pnet ( iCon, 'close' );
173               bClientConnected = false;
174
175             %% Server-Side Packet-Code 2, read trial packet of signal, event and classlabel data
176             case 2
177                aDataBuffer = pnet ( iCon, 'read', [1 2], 'double', 'network' );
178   %              aDataBuffer = pnet ( iCon, 'read', [1], 'double', 'network',[],'noblock' );
179               if ( ~isempty ( aDataBuffer ) )
180
181                 NUM_TRANSMITTED_SENSORS = aDataBuffer(1);
182                 NUM_TRANSMITTED_EVENTS  = aDataBuffer(2);
183
184                 % Index vector + classlabels + sensors + events
185                 NUM_SIGNALS_TOTAL = NUM_SIGNALS_META + NUM_TRANSMITTED_SENSORS + NUM_TRANSMITTED_EVENTS;
186                 aDataBuffer = pnet ( iCon, 'read', [NUM_SIGNALS_TOTAL TRIAL_LENGTH*SAMPLE_RATE], 'double',
                          'network' );
187
188                 if ( ~isempty ( aDataBuffer ) )
189                   % Add received packet(trial) to the total amount of data
190                   stData.aTriggers   = [stData.aTriggers   size(stData.aSignal,1)+1];
191                   stData.aSignal     = [stData.aSignal;
                            aDataBuffer(NUM_SIGNALS_META+1:NUM_SIGNALS_META+NUM_TRANSMITTED_SENSORS,:)'];
192                   stData.aTrueLabels = [stData.aTrueLabels aDataBuffer(1,1)];
193
194                     if ( TRAINING_WINDOW_SIZE == -1 )
195                         iStartIdx = 1;
196                     else
197                         iStartIdx = max ( iPacketNumber - (TRAINING_WINDOW_SIZE-1), 1 );
198                     end
199
```

```
200                     aCurTriggers = stdData.aTriggers ( 1, iStartIdx:size(stdData.aTriggers,2) );
201                     aCurCLLabels = stdData.aTrueLabels ( 1, iStartIdx:size(stdData.aTriggers,2) );
202                     iActNumClasses = -1;
203
204                       % Config OR-params
205                     if ( stdData.bFirstAttempt )
206                       iActNumClasses = NUMBER_OF_CLASSES;
207                       aActClasslabels = ALL_CLASSLABELS;
208                     else
209                       iActNumClasses = 2;
210                     end
211
212                     [iCurMinClasses, iSingleCountsBeforeOR] = GetClassCounts ( aCurCLLabels,
                                aActClasslabels );
213                     fprintf ( '\n## [%s] TCP-FLAG [%d], read packet [%d] [%s], true-label [%d] OptIdx[%d]
                                Trials %s\n', GetTimeStamp ( ), aBuffer(1), iPacketNumber,
                                num2str(size(aDataBuffer)), aDataBuffer(1,1), iOptIdx,
                                mat2str(iSingleCountsBeforeOR) );
214
215                     if ( TRAINING_WINDOW_SIZE > 0 )
216                       fprintf ( '## Win size [%d], tot packs [%d/%d] using [%d] triggers [%s]\n',
                                  TRAINING_WINDOW_SIZE, iPacketNumber, size(stdData.aTriggers,2),
                                  length(iStartIdx:size(stdData.aTriggers,2)),
                                  num2str(iStartIdx:size(stdData.aTriggers,2)) );
217                     end
218
219                     iPacketNumber = iPacketNumber + 1;
220
221                 else
222                   fprintf ( '## [%s] Received empty packet-buffer\n', GetTimeStamp() );
223                 end
224               end
225
226               otherwise
227               fprintf ( '## [%s] Unknown TCP-FLAG [%d] \n', aBuffer(1), GetTimeStamp() );
228             end %SWITCH CASE
229           else
230             fprintf ( '## [%s] Remote connection lost, listening again \n', GetTimeStamp() );
231
232             bClientConnected = false;
233           end
234
235         if ( bClientConnected && ( iCon ~= -1 ) )
236           bNextID = pnet ( iCon, 'read', 1, 'double', 'network','view','noblock' );
237             if ( isempty ( bNextID ) || bNextID ~= 2 )
238                bMorePackets = false;
239             end
240         end
241
242
243
244         end % WHILE (bMorePackets)
245                     tic;
246         if ( bClientConnected && ( iCon ~= -1 ) )
247                     %% Outlier rejection
248                     if ( ( iCurMinClasses >= MINIMUM_TRIALS_PER_CLASS ) && ( iCurMinClasses > (
                                iLastMinClasses + iOutRejOffset + TRIAL_STEP_SIZE ) ) )
249
250                       fprintf('\n##  Performing Outlierrejection... \n ');
251                       ORtime01 = tic;
252                       [cOut, aAllSigOutIdx, aOutCHIdc] = PerformReducedOR ( cOut, stdData.aSignal(:,
                                  NUM_TRANSMITTED_SENSORS-NUM_RAW_CHANNELS:NUM_TRANSMITTED_SENSORS),
                                  stdData.aTriggers, SAMPLE_RATE);
253
254                       caAllOutIdc{iPacketNumber} = aAllSigOutIdx;
255                       caAllOutCHIdc{iPacketNumber} = aOutCHIdc;
256                       caAllWeights{iPacketNumber} = w;
257                       caAllINI{iPacketNumber} = INI;
258                       ORtime02 = toc(ORtime01);
259                       fprintf ( '      OR time: %3f sec.,  Rejected [%d] outliers in [%d] trials %s \n',
                                  ORtime02, length(aAllSigOutIdx), iPacketNumber, mat2str(aAllSigOutIdx) );
260
261                       iOutRejOffset = 2;
262
263                     end
264
265                     % Reject data from training set.
266                     aCurTriggers(aAllSigOutIdx) = [];
267                     aCurCLLabels(aAllSigOutIdx) = [];
268
269
270                     [iCurMinClasses, iSingleCounts] = GetClassCounts ( aCurCLLabels, aActClasslabels );
271                     if ( ( IsAllTrue( iSingleCounts > 0 ) ) & ( iCurMinClasses >= MINIMUM_TRIALS_PER_CLASS )
                                & ( iCurMinClasses > ( iLastMinClasses + TRIAL_STEP_SIZE ) ) )
272     %% HERE STARTS THE GAAAAUUUUUUUDI
273                       traintime01= tic;
274
275                       for ( k = 1:size ( ALL_CL_COMBINATIONS, 1 ) )
276
277                           iCurClasslabels = ALL_CL_COMBINATIONS(k,:);
278                           % Return the indices for the minimum number
279                           % of available trials in both classes.
280                           aIdxCL1 = find ( aCurCLLabels == iCurClasslabels(1), iCurMinClasses, 'first' );
281                           aIdxCL2 = find ( aCurCLLabels == iCurClasslabels(2), iCurMinClasses, 'first' );
282                           aActIdx = [aIdxCL1 aIdxCL2];
283                           aActCLLabels = aCurCLLabels(aActIdx);
284                           aActTriggers = aCurTriggers ( aActIdx );
285                       end
```

```
286                     fprintf('## MinClasses = [%d] \n aActCLLables = [%s] \n aCurCLLabels [%s] \n
                               aActTriggers = [%s] \n aCurTriggers [%s]',...
287                         iCurMinClasses,mat2str(aActCLLabels),mat2str(aCurCLLabels),...
288                         mat2str(aActTriggers),mat2str(aCurTriggers));
289
290                     fprintf('      Start training... \n ');
291                     stData.ClassPos1 = aActTriggers(aActCLLabels == 7);
292                     stData.ClassPos2 = aActTriggers(aActCLLabels == 8);
293                     stData.ClassData1 = [];
294                     stData.ClassData2 =  [];
295                     stData.helpingHand = [];
296                     stData.model_csp_shrink = [];
297                     stData.all_scsp_filter =  [];
298                     stData.Class1counter = 0;
299                     stData.Class2counter = 0;
300                     stData.fb_counter = 1;
301                     stData.aSignal_csp = [];
302
303                     for current = 1:NUM_CHANNELS:NUM_TRANSMITTED_SENSORS-NUM_RAW_CHANNELS
304
305                     for CurrentTrial1 =  1:length(stData.ClassPos1)
306                         stData.helpingHand =[stData.helpingHand
                                   stData.aSignal(int64(stData.ClassPos1(CurrentTrial1)+...
307                             4.75*SAMPLE_RATE): int64(stData.ClassPos1(CurrentTrial1)+...
308                             7.75*SAMPLE_RATE-1),current:current+13-1)'];
309                     end
310                     stData.Class1counter = stData.Class1counter +1;
311                     stData.ClassData1(:,:,stData.Class1counter) = stData.helpingHand;
312                     stData.helpingHand = [];
313
314                     for CurrentTrial2 =  1:length(stData.ClassPos2)
315                         stData.helpingHand =[stData.helpingHand
                                   stData.aSignal(int64(stData.ClassPos2(CurrentTrial2)+...
316                             4.75*SAMPLE_RATE): int64(stData.ClassPos2(CurrentTrial2)+...
317                             7.75*SAMPLE_RATE-1),current:current+13-1)'];
318                     end
319                     stData.Class2counter = stData.Class2counter+1;
320                     stData.ClassData2(:,:,stData.Class2counter) = stData.helpingHand;
321                     stData.helpingHand = [];
322                     end
323
324
325                     for idx =  1:NUM_FILTERBANKS
326 %                     if iCurMinClasses >8
327
328                         stData.model_scsp=csp_train(stData.ClassData1(:,:,idx),...
329                             stData.ClassData2(:,:,idx),'standard');
330 %                     else
331 %
        stData.model_scsp=csp_train(stData.ClassData1(:,:,idx),stData.ClassData2(:,:,idx));
332 %                     end
333
334                         stData.model_csp_filter = [stData.model_scsp(:,1:3)
                                   stData.model_scsp(:,end-2:end)];
335                         stData.all_send_filter(:,:,idx) =stData.model_csp_filter;
336 %                       stData.all_scsp_filter = [stData.all_scsp_filter stData.model_csp_filter'];
337
338                     end
339
340                     for current = 1:NUM_CHANNELS:NUM_TRANSMITTED_SENSORS-NUM_RAW_CHANNELS
341
342                             stData.aSignal_csp = [stData.aSignal_csp
                                       stData.aSignal(:,current:current+NUM_CHANNELS-1)*
                                       stData.all_send_filter(:,:,stData.fb_counter)];
343                             stData.fb_counter = stData.fb_counter +1;
344
345                     end
346                     stData.fb_counter = 1;
347                     % Calculate power of CSP filtered signals
348                     stData.aSignal_csp_BP = (stData.aSignal_csp.^2);
349                     % Moving average
350                     a = 1;
351                     b = 1/SAMPLE_RATE*ones(1,SAMPLE_RATE);
352                     stData.aSignal_csp_BP_filt = filter(b,a,stData.aSignal_csp_BP);
353                    % Feature Extraction of the designated segements
354
355                    for current = 1:length(aActCLLabels)
356                         stData.segment =
                                   (stData.aSignal_csp_BP_filt(aActTriggers(current)+5.5*SAMPLE_RATE-1,:));
357                         X_trn =  [X_trn stData.segment'];
358
359                    end
360                    stData.modelRF = classRF_train(log10(X_trn'),aActCLLabels',ntrees,mtry,extra_options);
361                    stData.modelRF.CSP_matrices =  stData.all_send_filter;
362                    stData.modelRF.analysis_stuff.or = aAllSigOutIdx;
363                    stData.modelRF.analysis_stuff.true_cl = stData.aTrueLabels;
364                    stData.modelRF.analysis_stuff.iCurMinClasses = iCurMinClasses;
365                    X_trn = [];
366
367                traintime02 =toc(traintime01);
368                fprintf('      Training done. Took [%3f] sec -  Sending... \n', traintime02);
369                        if ( pnet ( iCon, 'status' ) > 0 )
370
371                            sendtime01 = tic;
372                            stData.modelRF.SaveMe = 1;
373                            stData.modelRF_serialized  = hlp_serialize(stData.modelRF);
374                            psize = size(stData.modelRF_serialized);
375                             pnet ( iCon, 'write', 50 );
```

```
376                         pnet ( iCon, 'write', size ( [stData.modelRF_serialized], 1 ) );
377                         pnet ( iCon, 'write', [stData.modelRF_serialized],'uint8' );   % Send Indices and
                                other Information
378                         sendtime02= toc(sendtime01);
379                         fprintf('      Sending DONE. Took [%3f] sec \n', sendtime02);
380
381                      end
382                      iLastMinClasses = iCurMinClasses;
383
384                      iOutRejOffset = 0;
385                     end
386          end
387
388        toc(wholeone)
389        end
390
391        % Wait for new client connection
392        if ( bServerRunning )
393          iCon = pnet ( iSock, 'tcplisten' );
394        end
395      end
396
397  fprintf('##############  SAVE Temporary. \n');
398  tic
399      save ( ['./rec/TMP_' sSubject '_' GetTimeStamp(2) '.mat'], '-v7.3' );
400
401  toc
402      display ( ' ' );
403    end
404
405    diary off;
```

Listing A.2: Optimizer- MATLAB implementation

A.3 Scripts for analysing the acquired Data

A.3.1 Calculation of the mean Accuracy

```
clear all;


color_vec = {'k', [0 0.4 0], [0.69 0 0], [0 0.4 0.4] , [0.85 0.85 0], 'm', 'b', [0.4 0.4 0.4], [0.54 0.27
       0],[0.25 0.25 0],[1, 0.36 0.14]};

 idx= 15

aData.true_vs_predicted.accuracy_peak = [];
aData.true_vs_predicted.accuracy_fb_mean = [];
aData.true_vs_predicted.accuracy_fb_median = [];

aData.true_vs_avg.accuracy_peak = [];
aData.true_vs_avg.accuracy_fb_mean = [];
aData.true_vs_avg.accuracy_fb_median = [];

figure
line ([3,3], [0,1], 'Linewidth', 3)
%
for idx = 1: length(data_folder_vec)-1

disp ('Initiating Fallout Analysis ...');

data_path = ['..\rec\fallout\' data_folder_vec{idx} '\*.gdf']

% Parameter

trial_length =  8; % sec.
before_cue = 3;
after_cue = 5;
number_training_lables = 10;
iCurclasslables =[ 7 8];
%% Loading data

disp('Loading Data');

[signals, header, events,files] =  gdf_multiread(data_path);

disp('done.')

%Insert zeros instead of NaNs
disp('Insert zeros instead of NaNs...')
for current_signal = 1: length(signals)
    tmp = signals{current_signal};
    tmp(isnan(tmp)) = 0;
    signals{current_signal} = tmp;
end
clear tmp
disp('done.')

%% Resample slow signals
disp('Resample slow signals...')
predicted_class_256Hz = resample(signals{14,1},16,1);
true_class_256Hz = resample(signals{15,1},16,1);
average_class_256Hz = resample(signals{16,1},16,1);
sys_sim_256Hz = resample(signals{17,1},8,1);
disp('done.')

%% classlables
disp('classlables...')
positions = [];
classlables = [];

for current=1:length(events.event_code)
    if events.event_code(current)==7
        positions=[positions,events.position(current)];
        classlables=[classlables,7];
    elseif events.event_code(current)==8
        positions=[positions,events.position(current)];
        classlables=[classlables,8];
    end
end

disp('done.')

disp('Accuracy...')
SR=events.sample_rate;
accuracy = NaN(1, SR*trial_length);



aIdxCL1 = find ( classlables == iCurclasslables(1), number_training_lables, 'first' );
aIdxCL2 = find ( classlables == iCurclasslables(2), number_training_lables, 'first' );
aIdxCL = [aIdxCL1 aIdxCL2];


acc_lables = classlables;
acc_lables(aIdxCL) = [];
acc_trigger = positions;
acc_trigger(aIdxCL) = [];
```

```
accuracy_all = NaN(length(acc_lables), SR*trial_length);
% TVP
%
for current=1:length(acc_lables)

    tmp = round(predicted_class_256Hz(acc_trigger(current)-...
        uint32(before_cue*SR):acc_trigger(current)+uint32(after_cue*SR)-1,:));
    accuracy(acc_lables(current) == tmp) = 1;
    accuracy(acc_lables(current) ~= tmp) = 0;

    accuracy_all(current,:) = accuracy;

end

accuracy_mean = mean(accuracy_all,1);
accuracy_std = std(accuracy_all,1);

accuracy_peak = max(accuracy_mean);
disp(['Peak accuracy:' num2str(accuracy_peak)])

accuracy_fb_mean = mean(accuracy_mean(1152:1920));
disp(['Mean accuracy over feedback period:' num2str(accuracy_fb_mean)])

accuracy_fb_median = median(accuracy_mean(1152:1920));
disp(['Median accuracy over feedback period:' num2str(accuracy_fb_median)])

fileID = fopen('results.txt','at');
if fileID ~= -1
fprintf(fileID,'%s & %.3f & %.3f & %.3f \r\n', subject{idx}, accuracy_peak, accuracy_fb_mean,
       accuracy_fb_median)
fclose(fileID)
end

aData.true_vs_predicted.accuracy_peak = [aData.true_vs_predicted.accuracy_peak  accuracy_peak ];
aData.true_vs_predicted.accuracy_fb_mean = [aData.true_vs_predicted.accuracy_fb_mean  accuracy_fb_mean];
aData.true_vs_predicted.accuracy_fb_median = [aData.true_vs_predicted.accuracy_fb_median
       accuracy_fb_median];

% Plot mean accuracy
hold all
grid on
disp('Plot mean accuracy...')
figure(1)
plot((1:length(accuracy_mean))/SR,smooth(accuracy_mean,45), 'Linewidth', 2, 'Color', color_vec{idx})
set(gca, 'XLim', [0 8], 'YLim', [0 1], 'FontSize', 14)
title('Hand versus Foot: Mean accuracy')
disp('done.')
xlabel('time [sec.]')
ylabel('accuracy')
hleg1 = legend(legend_subject,'Location', 'NorthEastOutside');

 end

disp(['True versus Predicted: Peak:' mat2str(mean(aData.true_vs_predicted.accuracy_peak))...
    ' Mean:' mat2str(mean(aData.true_vs_predicted.accuracy_fb_mean)) 'Median: '
          mat2str(mean(aData.true_vs_predicted.accuracy_fb_median))]);
```

Listing A.3: Calculation the mean accuracy over the trial period

A.3.2 Calculation Random Forest Analysis

```
%% Analysis of the FALLOUT RF Classifier %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

clc
disp ('##Initiating Fallout RF - Analysis ...');
data_folder_vec = {}
subject = {};

figure
hold all
%
 sub_idx =15;
%
  for sub_idx = 1:length(data_folder_vec)-1


path = ['../rec/fallout/' data_folder_vec{sub_idx} '/classifier/'];
% path = ('../rec/fallout/savetest_20140225/classifier/');

%% Here Starts the Gaudi!
disp('  Loading Data.');
files = dir([path '*.mat*']);

% All importance data per subject is saved in one COLUMN for each
% classifier. 15 columns means 15 classifiers were trained

for file_idx = 1:length(files)
  all_rf_cfr{file_idx} =   load([ path files(file_idx).name]);
```

```
31    importance_vec(:,file_idx) = all_rf_cfr{file_idx}.store_it.importance (:,4);
32    mintrials_vec(:,file_idx) = all_rf_cfr{file_idx}.store_it.analysis_stuff.iCurMinClasses;
33    oob_vec(:,file_idx) = all_rf_cfr{file_idx}.store_it.errtr(1000,1);
34  end
35
36  disp('Done!');
37
38  %EACH FILTER IS DISPLAYED IN A ROW OVER TPC
39  % FIFTEEN ROWS MEAN FIFTEEN FILTER!
40  disp('  Calculate Feature Importance.');
41  for current = 1:1:size(importance_vec,2)
42      counter = 1;
43      for idx=  1:6:size(importance_vec,1)
44         importance_bands(current,counter) = 1/6 * sum(importance_vec(idx:idx+6-1,current))  ;
45         counter=counter +1;
46      end
47
48  end
49
50  importance_bands =  flipud(importance_bands');
51
52
53  figure
54  imagesc(importance_bands)
55
56  set(gca,'YTick',1:15, 'YTickLabel',{' 35-40 Hz', ' 32-37 Hz',' 29-34 Hz',' 26-31 Hz',' 23-28 Hz',' 20-25
          Hz',' 17-22 Hz',' 14-19 Hz',' 12-14 Hz',...
57      ' 11-13 Hz','10-12 Hz','  09-11 Hz','08-10 Hz', '07-09 Hz','06-08 Hz'   });
58  set(gca,'XTick',1:length(mintrials_vec),'XTickLabel', mintrials_vec, 'FontSize', 13)
59  ylabel('Filterbanks')
60  xlabel('Trials per Class')
61  str_title = [ 'Subject ' subject{sub_idx} ': Feature Importance' ];
62  colorbar()
63  title(str_title)
64  disp('Done!');
65
66  disp('  Calculate OOB over TPC');
67  figure
68
69  plot(1:length(oob_vec), oob_vec)
70  set(gca,'XTick',1:length(mintrials_vec),'XTickLabel', mintrials_vec)
71  xlabel('Trials per Class')
72  ylabel('OOB error rate')
73  %
74
75
76  str_title_oob = [ 'Subject ' subject{sub_idx} ': OOB Error' ];
77  % figure
78  plot( 1:length(oob_vec),oob_vec', 'LineWidth', 2)
79  set(gca,'XTick',1:length(mintrials_vec),'XTickLabel', mintrials_vec, 'FontSize', 13)
80  title(str_title_oob)
81  xlabel('Trials per class')
82  ylabel('Out of the Box error')
83  oob.oob{sub_idx} = oob_vec;
84  grid on;
85
86
87  clear mintrials_vec
88  clear importance_bands
89  clear importance_vec
90  clear idx
91  clear oob_vec
92   end
```

Listing A.4: Feature importance maps and OOB estimate

A.3.3 Outlier calculations

```
1  %% Analysis of the FALLOUT RF Classifier %%
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3
4  clear all;
5
6  clc
7  disp ('##Initiating Fallout RF - Analysis ...');
8  sub_selector = 1;
9  % sub_selector = 0;
10 if sub_selector == 1
11 data_folder_vec = {};
12 subject = {};
13 naiv_stamp = { };
14 legend_subject = {};
15 else
16 data_folder_vec = {};
17 subject = {};
18 legend_subject = {};
19 naiv_stamp = { };
20 end
21 legend_text = {'mean'};
22
23
```

```
24  figure
25  hold all
26  line([0 4],[  7.6667   7.6667],'Color', [1 0 0,],  'Linewidth', 3, 'LineStyle', '--');
27
28
29
30
31  %
32    for sub_idx = 1:length(data_folder_vec)
33
34
35  path = ['../rec/fallout/' data_folder_vec{sub_idx} '/classifier/'];
36  % path = ('../rec/fallout/savetest_20140225/classifier/');
37
38  %% Here Starts the Gaudi!
39  disp('  Loading Data.');
40  files = dir([path '*.mat*']);
41
42  % All importance data per subject is saved in one COLUMN for each
43  % classifier. 15 columns means 15 classifiers were trained
44
45  for file_idx = 1:length(files)
46    all_rf_cfr{file_idx} =   load([ path files(file_idx).name]);
47    importance_vec(:,file_idx) = all_rf_cfr{file_idx}.store_it.importance (:,4);
48    mintrials_vec(:,file_idx) = all_rf_cfr{file_idx}.store_it.analysis_stuff.iCurMinClasses;
49    oob_vec(:,file_idx) = all_rf_cfr{file_idx}.store_it.errtr(1000,1);
50    or{:,file_idx} =all_rf_cfr{file_idx}.store_it.analysis_stuff.or;
51  end
52
53  all_or(:,sub_idx) = numel(or{end});
54  bar(sub_idx,numel(or{end}))
55  clear or
56
57
58
59    end
60
61  set(gca, 'Xtick',1:3, 'Xticklabel', subject,'FontSize', 14)
62  str_title = [ 'Outlier Rejection' ];
63
64  title(str_title)
65  disp('done.')
66  xlabel('Subjects')
67  ylabel('Rejected Outliers')
68  hleg1 = legend(legend_text,'Location', 'NorthEast');
```

Listing A.5: Calculation of the Outliers and Graphical representation