SimBCI-A framework for studying BCI methods by simulated EEG

simBCI - A framework for studying BCI methods by simulated EEG

Jussi T. Lindgren[1], Adrien Merlini[2], Anatole Lecuyer[1], and Francesco P. Andriulli[2]

[1]Univ. Rennes, Inria, IRISA, CNRS
[2]CERL @ IMT-Atlantique

January 25, 2018

Abstract

Brain-Computer Interface (BCI) methods are commonly studied using Electroencephalogram (EEG) data collected from human experiments. For understanding and developing BCI signal processing techniques real data is costly to obtain and its composition is apriori unknown. The brain mechanisms generating the EEG are not directly observable and their states cannot be uniquely identified from the data. Subsequently, we do not have generative ground truth for real data.

To allow studying BCI signal processing methods in controlled conditions, we propose an object-oriented framework called *simBCI*. The frawework is for generating artificial BCI data and to test classification pipelines with it. Parameters of interest on both data generation and the signal processing side can be iterated over to study the effects of different combinations. The architecture allows affordable exploration of how BCI signal processing approaches behave in different conditions by enabling precise control over the generative process. The proposed system does not intend to replace human experiments. Instead, it can be used to discover hypotheses, study algorithms, to educate about BCI, and to debug BCI signal processing pipelines.

The proposed framework is modular, extensible and freely available as open source[1]. The only requirement is Matlab.

[1]Downloadable from `http://gitlab.inria.fr/sb/` . Until the paper has been accepted and the code can thus be publicly released, please use the login 'sbreviewer' and password 'Investigator35b22--' to obtain the code.

1 Introduction

Brain-Computer Interfaces (BCIs) based on Electroencephalogram (EEG) attempt to translate measurements of the user's brain activity into commands for the computer. To obtain a discrete command, the BCI methods process and classify segments of the EEG signal. Unfortunately the current BCIs leave a lot to be desired in terms of accuracy [1, 2, 3]. Why is this the case? Studying this question from the signal processing viewpoint is difficult for several reasons. One is the cost of running large-scale human experiments for statistical validity, and the other is the nature of the obtained EEG data. As the generative sources in the brain cannot be uniquely identified from the EEG [4], we lack ground truth regarding the components of the data. In these conditions, we do not know what kind of performance an optimal BCI could have on such data. It is even hard to be sure if one method is better than another (see e.g. [5]). It is also difficult to assess how the different elements of the BCI session contributed to the obtained results. How are the results affected by the user, the equipment, and the signal processing? Which properties of these elements had what effects?

In this paper, we propose to alleviate the study of signal processing methods in BCI by an open source framework called *simBCI* for simulating some central parts of BCI experiments. In particular, the framework can generate simulated BCI data and execute signal processing (or classification) pipelines. Parameters in both generation and classification can be varied, and the framework can provide aggregates for quantities of interest such as prediction accuracies per parameter configuration. The intent of the framework is to allow studying the behaviour of the model BCI systems with different configurations and parameters, and to numerically investigate how different generative or discriminative parameters affect the results such as accuracies and estimated model parameters. If the modelling assumptions hold, similar behavior may be encountered in real circumstances.

The simulation framework we propose is conceptually situated halfway between mathematics and human experiments. We do not propose it to replace either. The benefit of simulation is that it can provide affordable hypothesis discovery and insight, while only requiring explicit models of the systems under study and some computational power. For example, simulation may suggest that one method is more resistant to a specific type of noise or artifact, or less affected by the positions of the discriminable signal sources in the brain volume. Such discovered hypotheses can then be further studied with real experiments or mathematical analyses. Another possible use-case is to illustrate the behavior of the used models for pedagogical purposes. The student can control the simulator and observe the input-output

relationships before and after different transformations of the signal. This can be done both during the data generation, and the signal processing. As yet another use-case, the generated data can be used to debug existing BCI systems: easy data can be generated that matches the modelling assumptions of the BCI systems' signal processing. If the system does not process such data as expected, an issue has been discovered. Further, if the user provides different head models and/or parameter sets, multiple datasets can be generated. In principle these datasets could be used to study behavior of techniques such as deep learning that may be able to benefit from a big data approach. However, we do not currently provide such learning algorithms in the system.

The presented framework is open source and available for free. We warmly welcome extensions and contributions to improve it. Due to the modular design, new components such as signal and noise models, head models, forward models, inverse algorithms and other signal processing plugins can be easily added to the system to improve its realism or to make it more applicable to study specific questions. Similarly to other Matlab-based systems, extensions can be simply added by depositing new scripts to the directory hierarchy. As the source code is hosted on a flexible and open git-based system, the community enthusiasts can easily clone their own versions of the framework and submit pull requests if they desire some components to be included in the upstream version.

The rest of this paper is organized as follows. We begin in section 2 by a more detailed description of the challenges that simulation can help with. We describe previous work in section 3. In section 4, we present an overview of the proposed architecture. In section 5 we describe how the framework models data generation and in section 6 we present the conventions used for signal processing. Then, section 7 shows how the generation and the processing components are put together to form a BCI simulator. Section 8 shows a few illustrative uses of the system with discussion in section 9. Finally, section 10 concludes.

2 Why simulation?

Since the introduction of computers, simulation has become an increasingly important technique in research and development in a multitude of fields [6], ranging from manufacturing [7] to brain research [8]. Although real experiments are ultimately needed for testing and verification, simulation is useful in obtaining hypotheses about the studied systems when human experiments and analytical studies are challenging and costly. In the scope of

BCI, we have identified three different reasons why studying signal processing techniques can be difficult with real data.

1. The cost of human experiments. A legion of different BCI signal processing pipelines have been proposed, typically with many parameters [9]. If some signal processing technique is promoted based on experiments with only a few subjects, there is a significant risk that these results are a statistical accident, unless very strict statistical controls are used. Large studies are to be preferred. Such statistically valid experiments require orchestrating a large number of subjects to perform long and fatiguing BCI sessions. The subjects have to be instructed, prepared and set up with the recording equipment. To attain successful BCI control, the users may also require preliminary training sessions. An overview of the work involved can be found e.g. in [10, 11, 3].

2. Problems in controlling the data generation. Due to different physiologies, mental strategies and possibly other causes, EEG signals generated by different users have different properties (e.g. in motor imagery [12, 13, 14]). As adapting the signal processing to these user-specific characteristics improves the classification accuracy, understanding these differences and their effects can be seen to be directly relevant for BCI development. In real experiments, some properties are either totally outside the control of the experimenter (such as the user anatomy, location of different functional areas in the brain, and electrical propagation through the tissues) or difficult to control and verify (used mental strategies). The knowledge of anatomical and functional details may be available in medical circumstances, but such information is not usually available in other use-cases.

3. Lack of ground truth. The ground truth regarding the signal generation cannot be easily obtained for real data. This is partly due to the information-losing physiological volume conduction process that blends, mixes and dampens the electrical activity when it propagates through the brain, the skull and the scalp before it can be measured by EEG [15]. Since we do not have direct access to what happened between the user receiving some instructions and the eventually observed EEG recording, it is difficult to determine whether a particular result is due to the user's skill, physiology, artifacts, noise, bugs in the system, particular choices in the signal processing, amount of training data, or perhaps nonstationarity of the brain activity.

In simulations, these issues can be circumvented. The experiments can be as large as the available computational power and time permits, and all the various parameters can be controlled up to the capability of the researcher to reasonably model the phenomena of interest. Since the data is generated with an explicit specification, we know exactly what is in the data, and hence we have access to ground truth on all the levels that we model in the

generation. Ultimately, the obtained hypotheses and intuition can be further studied with real experiments.

3 Previous work

Simulating EEG data has a long tradition. The generative models proposed for EEG can be used to hypothesize about the mechanisms behind the measurements, but also to synthesize simulated data [16, 17, 18, 19]. Simulated EEG data is commonly used to study EEG forward and inverse models in the context of source localization (e.g. [20, 21, 22]) and to test blind source separation methods [23]. It can be also used to study connectivity measures [24]. Several software packages exist that can generate synthetic EEG data. These include the Brainstorm package [25], the Fieldtrip toolbox [26], the SIFT toolbox [27] and the BESA Simulator[2]. Of these packages, Fieldtrip provides a few different generators for EEG trials, and the SIFT toolbox can generate data using autoregressive models. In principle these approaches could be customized by an advanced user to generate multiclass data for BCI testing purposes. However, this may require significant expertise from the part of the user, for example understanding of autoregressive modelling for SIFT.

In the scope of BCI, some previous simulation work exists. One possible approach is to simulate the predictions of the classifier [28, 29, 30] in order to study how usable an application would be when it's controlled with an inaccurate BCI. Another way to use simulation is to apply real recordings in an offline manner [31]. For example, the standard approach of testing BCIs by cross-validation can be considered simulation. In a real BCI session, the prediction errors made by the classifier may distract the user and hence provide a feedback effect. This effect is not modeled by offline cross-validation.

Simulated BCI data has been proposed before. One application is to attempt to use artificial data to complement real data for training the signal processing models [32]. Artificial datasets have also been used in at least one competition: the BCI Competition IV featured simulated motor imagery data [33]. The competitors were not informed beforehand that a particular dataset was made synthetically. Our framework contains the first public, open source re-implementation of the used generative technique. A detailed description of the approach can be found elsewhere [33].

Since the proposed framework not only simulates EEG data, but can also perform offline BCI signal processing and classification, it is in order to briefly explain the main difference of the proposed framework to the well-known BCI

[2] `http://www.besa.de/`

software platforms (for overview, see e.g. [34, 35]). The proposed framework is mainly intended for offline study of model behavior. Researchers interested in running realtime BCI experiments can consider e.g. OpenViBE [36] or BCI2000 [37]. Although our framework can perform signal processing on real data if the data is converted to the framework's conventions, a more extensive collection of signal classification components can be found e.g. in BCILAB [27]. For EEG data analysis, EEGLAB [27] and Fieldtrip [26] are classical choices. At present, our framework can carry out the signal processing in BCILAB if requested, and optionally export the data to EEGLAB for visualizations.

4 Framework overview

The proposed framework aims to simulate BCI-like data in a controlled and modular fashion and allow BCI signal processing pipelines to be constructed and tested in the framework as well. These pipelines can use machine learning techniques and/or inverse modelling as part of their operation if desired. For convenience, the framework provides a mechanism to test the effects of varying sets of parameters, both on the generation and the signal processing side. In the following, we provide a high-level overview of the framework. For more technical description, we invite the reader to look at the user documentation supplied with the framework.

The framework is designed with the goal that all the parameters of interest should be specifiable from a single high-level script. The various submodules that are used should fill in default values if some parameters are not given. The framework itself is agnostic about most of the parameters. They are simply provided to the submodules that recognize them. This convention allows the experimenter to see at a glance what the framework is requested to do and specify the experimental parameters from a single location.

The framework consists of three main classes as shown in Figure 1: The first is the **Generative model** that constructs simulated experiment timelines (sequences of event markers) and simulated EEG-like data using random generators[3]. The second class implements the BCI signal processing and classification. We call each realization of this class a **pipeline**. A pipeline is typically intended to be calibrated using the training data produced by the generator and tested by an independently generated test dataset. Finally, the **BCI Simulator** class is a convenience tool that can iterate over sets of parameters. It can also perform repeated sampling with the same parame-

[3]The framework is not tightly tied to EEG, and could in principle be modified for other streaming data.

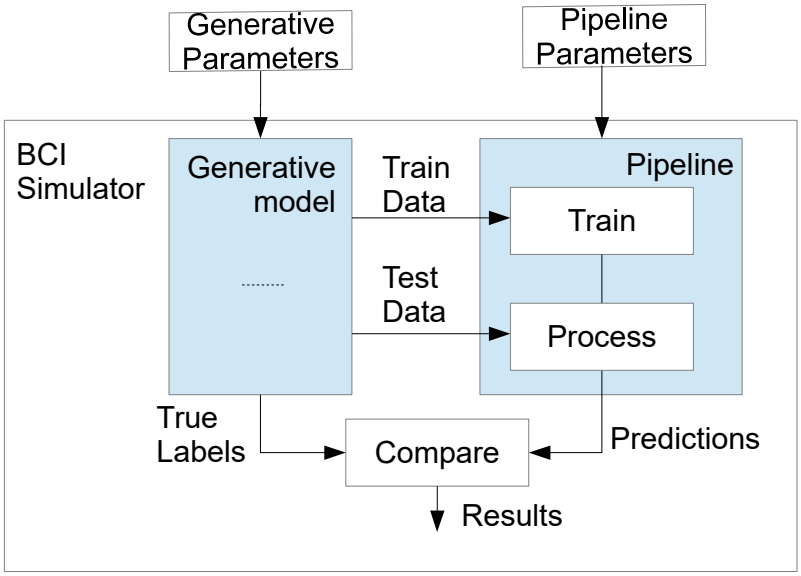


Figure 1: Data flow in the framework between the main components. Different parameter specifications define how the data is generated and what methods the signal processing pipeline uses.

ters in order to smooth out statistical variations in the results. In the end, the simulator computes prediction accuracies by comparing the pipeline outputs to the known ground-truths from the generator. As part of its design, the BCI simulator does not let the tested pipeline see the parts of the data that would not be available in real BCI experiment (e.g. trial labels, cortical sources, etc). Note that the generator and the pipeline classes can also be used separately. For example, the data generator can be used to simulate test data for some third-party BCI system without relying on the simulator or the pipelines.

We now turn to describe the main modules of the framework in more detail.

5 The generative model

The default signal generation module in the system follows the common linear superposition model [4, 38]

$$\mathbf{X} = \mathbf{AS} + \mathbf{N}, \tag{1}$$

where $\mathbf{X}$ is an [electrodes $\times$ samples] matrix of measured EEG observations over time, $\mathbf{A}$ is the *leadfield matrix* modelling the volume conduction and $\mathbf{S}$ is a [sources $\times$ samples] matrix of source activities in the volume. The matrix $\mathbf{S}$ can include both signal and noise components. $\mathbf{N}$ is a matrix of surface noise. The rows of $\mathbf{X}$ correspond to individual surface electrodes and the rows of $\mathbf{S}$ to source dipoles in the volume. The electrodes and the sources are assumed to have 3D coordinates in relation to a *head model* associated with the leadfield. The leadfield matrix coefficients encode the model of the electrical propagation from the sources to the surface. Depending on the used level of detail, such leadfields may vary from single-sphere models to physiologically realistic, subject-specific models with different compartments and their conductivities. For details, see e.g. [4, 38].

Due to the linear superposition model, the different data components can be generated both in the volume (in $\mathbf{S}$) and on the surface (in $\mathbf{N}$). The framework by default mixes the components on the surface. Although it would be possible to work directly with matrices $\mathbf{S}$ and $\mathbf{N}$, the dimensions of these matrices and the appropriate indexing depends on the used head model. Subsequently, their manipulation can become tedious and error-prone. For this reason, the framework provides a higher-lever interface to specify the signal components. With this interface, each signal component is assumed to have three properties: *when*, *what* and *where*. Their interpretation is intuitive. The *when* specifies the event times and durations on a timeline, *what* specifies the signal content that is introduced to the EEG by these events, and *where* is the placement of this activity either in the volume or on the surface. This is illustrated in Figure 2. Each component also has a power parameter (or Signal-To-Noise Ratio, SNR) which we have omitted from the figure for clarity. The whole process can be alternatively seen as *rendering* that realizes a concrete multichannel signal from a more abstract timeline.

Note that in the framework, the different signal components are, by default, independent of each other. The respective generators do not see the parameters or the outputs of the others. To introduce dependencies, the user can implement a monolithic generator that internally handles them. Alternatively, the user can design a timeline that has dependencies between the events.

To give an example of the workflow, the investigator (user) first specifies the desired event generators and their parameters (or even the actual event timeline), specifies signal generators that react to these events, and declares where each type of activity should occur with relation to the head model. The framework uses the head model to obtain the positions and indexes of the relevant dipoles in $\mathbf{S}$, and uses the associated forward model to project the volume data to the surface using eq. 1. With this approach, the head models

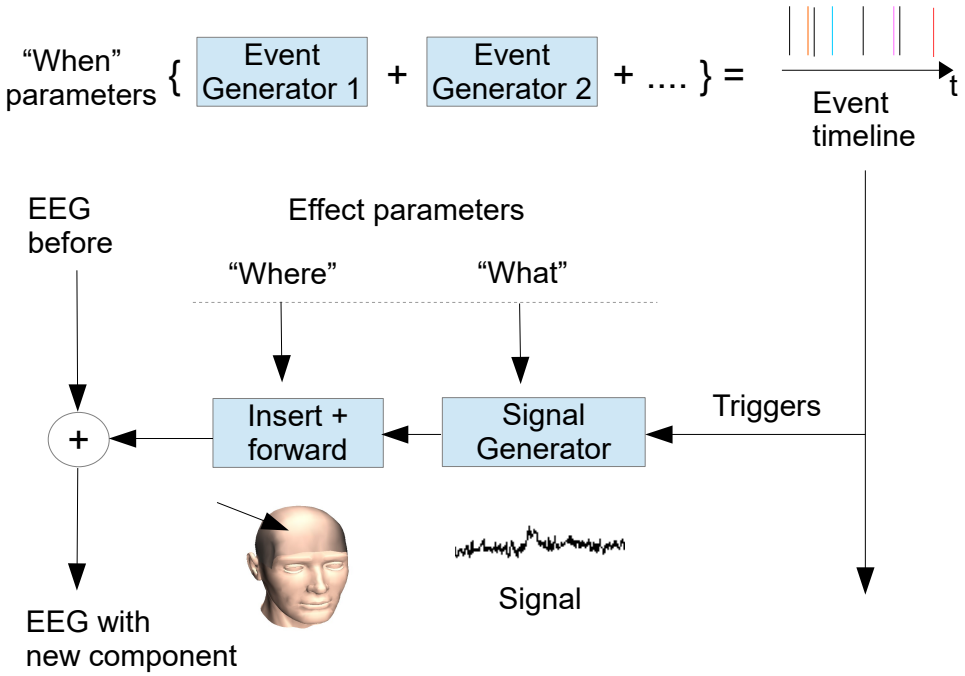


Figure 2: EEG signal generation. First, one or more event generators are used to make a timeline of discrete events with durations. Then, different generators are specified to react to the events. Each generators' output is then inserted to the signal as requested. In the figure we show only one signal component generation for clarity, even though the user can mix and match any amount of such components and provide new generators.

can be changed and the scripts remain interpretable as they are defined on a more semantical level. Geometrical information is encapsulated in the head model. The user can either use the example head models provided with the framework, provide their own, or assemble a head model with the assistance of third-party packages such as Fieldtrip [26] or OpenMEEG [39].

The proposed abstraction allows relatively easy adaptation of generative recipes from one type of BCI to another. For example, let us assume that two-class SSVEP [40] and Motor Imagery [41] experiments have similar noise profiles but different origin and nature of the signal part. Then, it suffices to change the 'what' and 'where' parts of the specification: going from Motor Imagery to a simple SSVEP model, the 'what' part is changed to a function that generates a specific SSVEP response frequency depending on the class of the trial, and the 'where' part is changed from left and right motor cortices to cover a region of the occipital lobe.

The framework provides heuristic methods to approximate certain locations of interest from the used head model geometry. Specific keywords such as *eyes*, *occipital lobe* and *left and right motor cortex* are recognized by the *heuristic where* function and use simple deterministic selection rules such as 'eyes are approximated by two dipoles that are about halfway in z, in front in x, and symmetrically about 1/4 skull-width from the y midline on both sides'. If accurate anatomical knowledge and cortical segmentation is provided in the head model, this can be used instead of the heuristic approach, simply by using an *exact where* function that refers to the head model.

The framework conventions also suggest that each callable function implements an optional visualization. By enabling the visualization, the function itself can illustrate the data it generates in a way that is suitable for such data. Additionally, the framework provides some simple visualizations for aggregated or surface signals.

Figure 3 shows the used concepts and some of their options available at the time of writing.

5.1 Data generation example: BCI Competition IV

We make the previously introduced concepts more concrete with a motor imagery example. We follow the technique used to create one of the datasets in the BCI Competition IV [33]. The proposed framework includes the first public re-implementation of the generator.

A specification to make data in the style of BCI Competition IV is shown in Figure 4. As can be seen, the specifications in the framework are either lists or composed of *key,value* pairs, where the value can also be a list of a function and its parameters. The parameters are passed to the functions

Component	Options
'where'	heuristic: eyes, occipital lobe, left and right motor cortexes, cluster of K dipoles exact: from the head model, whole surface, whole volume
'what'	Gaussian and pink noise, eye movements [33], eyeblinks [33], noise with spectral dependencies[33], single frequency noise (e.g. 50hz/60hz), beta desynchronization [33], P300 template, SSVEP frequency responses
'when'	Trial generator supports trial length, rest length, burn-in, class count and class ordering parameters. Random event generator is also available.
Examples provided	Motor Imagery [33], elementary SSVEP [40], elementary P300 [42]
Head models	9417x247 MRI-based, constrained/nonconstrained dipole orientation, sphere

Figure 3: Various options provided for data generation. User-provided options can be alternatively used.

that handle them.

The specifications in Figure 4 make it apparent how to change the various parameters such as the timeline of the experiment and the used head model without modifying the actual signal generators. The head specification declares the forward model used to map volume data to the surface and to provide the positions of the dipoles for the signal generation. The timeline specification contains the parameters related to synthesizing the experiment timeline. The effects specification concerns the actual activity simulated. Here, four data generators in total are specified and called in sequence. The first two simulate beta desynchronization at 12 Hz. The framework is instructed to insert these activities to the left and right motor cortex dipoles, heuristically localized using the specified head model. The next directive requests the generation of artifact noise originating from the eyes, and finally generic noise activity to be placed on the surface. The framework projects the volume data to the surface using the specified head model and then merges the different parts of the data by linear superposition with weighting chosen to give the desired SNR (on the surface).

Figure 5 illustrates some visualizations related to the generative specification we just described.

6 EEG classification pipelines

In the presented framework, a signal processing chain is called a *pipeline*. A pipeline can be calibrated with training data, and used to process new data. Pipelines are made using one or more modules called *processors*. These

```
headParams = {'filename', './leadfield.mat'};

timelineParams = { 'samplingFreq', 200, 'eventList', { ...
  {'when', {@when_trials, 'events',{'left','right'}, ...
            'numTrials',10, ...
            'trialLengthMs',4000, 'restLengthMs', 2000, ...
            'trialOrder', 'random', 'includeRest', true}}, ...
  {'when', {@when_random, 'events',{'eyeblink'},'eventFreq',0.1}}} ...
};

effectParams = { ...
  {'SNR', 1.0, 'name', 'signalLeft', 'triggeredBy', 'left', ...
   'what',  {@gen_desync, 'centerHz',12,'widthHz',1,'reduction',0.5}, ...
   'where', {@where_heuristic, 'position','rightMC'} }, ...
  {'SNR', 1.0, 'name', 'signalRight', 'triggeredBy', 'right', ...
   'what',  {@gen_desync, 'centerHz',12,'widthHz',1,'reduction',0.5}, ...
   'where', {@where_heuristic, 'position','leftMC'}}, ...
  {'SNR', 0.1, 'name', 'blinks', 'triggeredBy', 'eyeblink', ...
   'what',  @noise_eyeblinks, ...
   'where', {@where_heuristic, 'position','eyes'}}, ...
  {'SNR', loop_these([0.005, 0.001, 0.01]), ...
   'name', 'noise', 'triggeredBy', 'always', ...
   'what',  {@noise_spectrally_colored, 'subType','fake', ...
             'strength', [1.0 0.5, 0.3]}, ...
   'where', {@where_whole_surface}} ...
};
```

Figure 4: Simplified specification generating 2-class motor imagery data resembling the approach of the BCI Competition IV. The timeline specification has an event generator to make a basic two class timeline and an eyeblink event generator. The framework then triggers signal generators to react to these events. For example, the 'right' class event triggers 'left' motor cortex desynchronization in the above specification. The parameters that follow the functions specified with @ are simply arguments to those functions. In the last signal component, the directive 'loop_these()' tells the BCI Simulator to expand this specification to 3 new ones, each with different level of surface noise SNR.

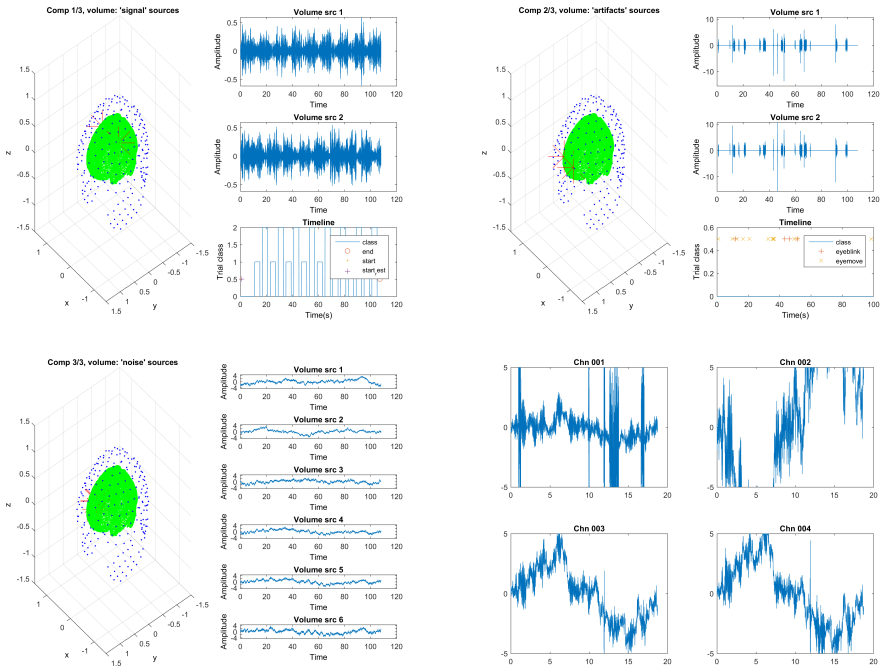


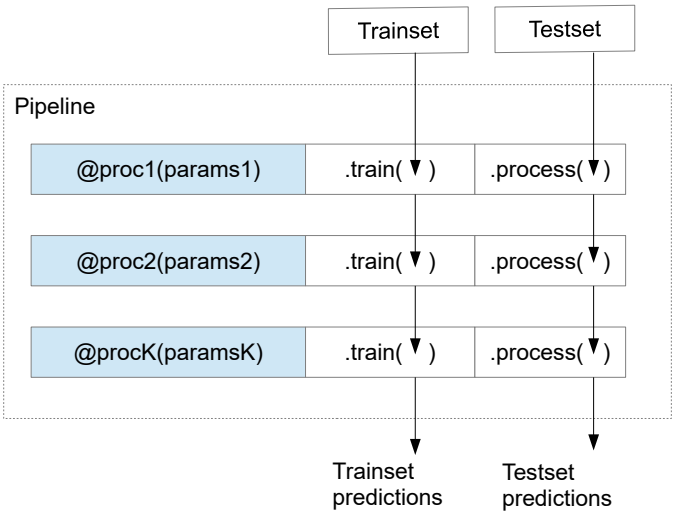
Figure 5: Components of a generated motor imagery signal as visualized by the framework. Top left, the discriminable signal from two desynchronization generators combined to one display. Top right, eye artifacts. Bottom left, the noise. Note that since the noise originates from the whole volume, only the 6 first dipoles are shown. Bottom right, first 15 s of surface data for the 4 first electrodes. The beta desynchronization generators are driven by the timeline shown, whereas the eye artifact generator reacts to random events (not shown). The noise generator is simply always active. After generation, these signals will be inserted to the places that were selected by the heuristic 'where' function and projected to the surface using the leadfield of the head model. Note how the eye artifacts are stronger than the mixture of the two other components in the surface data.

processors are used in the order they are specified in a list such as the one shown on the bottom of Figure 6. Separate specifications are not required for calibrating the pipeline and processing data with it, and neither separate processors are needed. Instead, the pipeline and all processors are classes that must provide 'train()' and 'process()' member functions. A pipeline is essentially constructed by feeding it training data with the 'train()' call, and data transformation results are obtained using the 'process()' call for some set of data. If the pipeline ends with a classifier, it is expected to output probability vectors, which can then be seen as just another data transformation.

With this design, only a single specification is needed for train and test. It avoids potential discrepancies between train and test runs. Yet the chosen approach can support components that do not need training (such as usual temporal filters, fixed arithmetic, etc) as well as machine learning and statistical components. The data-independent classes simply do nothing on train and just process the data when processing is requested. The classes that construct a model do this during the train() call and then encapsulate the estimated model inside the constructed class object. When a pipeline is calibrated, it calls the train function of each processor of the specification sequentially. The first processor gets the original dataset, and the second processor gets the data as it is after having been processed by the first processor, and so on. The processing of new data is performed in the same order. This is illustrated on the top of Figure 6. In the conventions of the framework, a pipeline is ultimately expected to return a class probability vector for each trial in the given dataset. However, this is not strictly necessary: a pipeline can return an arbitrary transformation of the data. Predictions are only used by the BCI Simulator class that compares the results to the trial labels. Note that the framework is in general agnostic to what the processors pass to each other, it just assumes that the processors are specified in an order that the next processor is able to consume the output of the previous processor.

Figure 6 illustrates also a pipeline defined with parts provided with the framework. It implements a classic CSP-bandpower based pipeline for classification of Motor Imagery [43]. In the specification, each processor is followed by parameters specific to that processor. It's worth noting that the pipeline could also be implemented as single monolithic class, if the described processing stages were carried out inside the class code. When implementing a new pipeline, the designer has to choose the preferred level of modularity.

Figure 7 lists processors bundled with the framework. The bundled set is meant to be illustrative and it is not intended to implement the full extent of the current BCI knowledge. Rather, the processors were written to sup-




```
pipeLDA = {'name','csp-bandpower-lda', 'processors', { ...
  {@proc_bandpass_filter, 'freqLow',8,'freqHigh',30}, ...
  {@proc_csp, 'dim', 2, 'tikhonov', 0.5, 'shrink', 0.5}, ...
  {@proc_power_transform, 'logFeats',true}, ...
  {@proc_normalize}, ...
  {@proc_lda} } };
```

Figure 6: Top. Schematic for a signal processing pipeline, expanding the Pipeline class of Figure 1. For efficiency, a train() call is not required to return any data. In that case, the framework simply calls the corresponding process() function with the training data to obtain input for the next processor (not shown). Bottom, an example of pipeline specification. The specification declares the usual CSP-Bandpower LDA pipeline (e.g. [43]) using 5 different processors in a sequence. Each processor is constructed with its own parameters.

Type	Variants provided
Classifiers	LDA [44], liblinear [45], MD Classifier (as in [46]), fixed threshold, BCILAB bridge [27]
Filters	Temporal filter, bandpass filter, BCILAB
Feat. extraction	CSP [43], inverse transforms (MN & WMN [38], sLORETA [47], ...), time/frequency features, ICA [48], time interpolation
Other	Correlation feature selector, normalization, downsampling
Simple DSP	Squaring+log, identity, BCILAB

Figure 7: Basic processors provided in the framework to build pipelines. New processors can be inserted by adding new Matlab classes to the directory tree. The classes just need to conform to the interface conventions of the framework.

port specific studies regarding CSP-Bandpower-LDA and inverse modelling. Hence, these modules suffice to illustrate how to design pipelines using the conventions of the framework. A user wishing to quickly evaluate a larger set of state of the art is suggested to use the 'BCILAB' processor provided in the framework. This way, any processing available in BCILAB can be used. The framework by default provides specifications to construct the CSP/Bandpower/LDA pipeline (e.g. [43]) as well as a few pipelines based on inverse models: One is inspired by Cincotti et al. [49] and the another is after Edelman, Baxter and He [46]. The user is invited to look into the package archive to find these and other example specifications.

7 BCI simulation

Artificially generated data and signal processing pipelines provide the central components to carry out BCI simulations. In the proposed framework, we also include a third central component called the BCI Simulator. This is a convenience tool used to repeatedly call the data generation and the pipeline training and testing, to compare the results to ground truth, and to aggregate the results. By default, prediction accuracies are computed and stored. Given the definitions for generation and the pipelines, the Simulator can perform repeated experiments by resampling new sets of data while keeping the sampling parameters the same. This way, better statistical characterization of the results related to each parameter condition can be obtained: enough iterations allows the empirical estimates of the pipeline accuracy and variance to converge and reduces the chance that some particular result was due to a statistical accident.

The system allows testing the effects of parameter changes by providing

a simple construct for the purpose. The specifications (such as those shown in Figures 2 and 6) controlling the system can contain statements which essentially instruct the Simulator to run multiple experiments while taking the parameter values from given sets. For example, an SNR parameter for some signal component could be specified as a range of values to test, as in Figure 2. In the figure, the simulator will construct 3 experiments where each has a different value of the SNR but keeps the other parameters equal. A similar mechanism is available in BCILAB [27] for classification. In our framework, the loop requests can be used both in the generating and the pipeline specifications. It is also possible to have multiple loop requests. Then the total number of train/test runs in the experiment is a product of the sizes of the combinations times the number of repetitions.

Note that for data generation, the parameters can also be different between the train and tests sets. The only compatibility requirement between the two sets is that the pipelines estimated with a training set must be able to return results comparable to the ground truth when they are provided the test set. For example, the specifications can differ in the number of trials or the nature of noise. Also, if inverse models are used in the pipelines, different head models can be used in the data generation and the pipeline definitions. This can simulate the effect that in practice we rarely have a perfect head model of the BCI user in question. On the other hand, the structural parameters such as the number of sources and electrodes should stay the same, unless the pipeline processors are modified to handle such differences, for example by interpolation.

8 Example simulations

We presume a typical use of the framework is to simulate experiments by repeatedly generating data with different characteristics (parameters) while possibly modifying the pipeline parameters at the same time and then analyze the resulting classification accuracies. This allows empirical discovery of what is easy and what is difficult for different algorithms. For example, signal generation parameters such as artifact occurrence frequency or SNR could be modified while observing how this affects the BCI pipeline prediction accuracy. Experiment parameters can also be modified, for example the number of trials, trial length and the sampling rate. On the pipeline side it is possible to change everything from algorithm parameters (e.g. regularization, filter bandwidths, CSP dimensions, etc) to the algorithms themselves.

The signal data used in the experiments described in this section has been generated using forward models based on a standard three-layer mesh (scalp,

skull, brain) obtained from processing MRI data with FieldTrip [50]. The three layers were assigned respective normalized conductivities of 1, 1/15 and 1 (as in [51]) and were composed of up to 46 000 triangles for the high resolution cases. The generated models map 9417 cortical dipoles oriented along the normal to the cortical surface to 249 electrodes. An ajoint double layer formulation has been used to compute this mapping.

Eyeblink frequency vs eyeblink power. As an example, Figure 8 shows how the common CSP/LDA pipeline prediction accuracy behaves when the eyeblink artifact frequency and its SNR are changed. The data generation was following the BCI Competition IV style of Motor Imagery simulation using the model described earlier. In this case we replaced the data driven noise of the original approach [33] with volumetric pink noise to ensure the results are not due to a particularities of a specific EEG recording that is normally used to obtain the method's spectral dependency model. The SNR of the pink noise was calibrated to give 80% pipeline prediction accuracy without any eyeblinks.

After the detrimental effect of the eyeblinks is observed, the experimenter could proceed e.g. by studying if the pipeline regularization parameters would help against the eyeblinks, or how much more training data would be needed (in principle) to compensate for them. In the framework, this would be just a matter of specifying ranges for these additional parameters of interest and let the computer perform the experiment. Alternatively, the researcher might attempt to implement an eyeblink removal technique and examine how well it performs, or route the data to some more robust signal processing technique. After satisfying intuition and solution to the problem has been obtained, real experiments could be carried out to validate the hypotheses formed with the simulated experiment.

Suppose that the experiment described above was done with real subjects. They would be asked to modulate the strength and frequency of their eyeblinks in addition to performing the challenging task of motor imagery at the same time. How much time would such a study take? In total, to draw the plot of Figure 8, we tested 60 parameter combinations with 25 resampled repetitions for each combination. This means 1500 simulated EEG sessions consisting of generating a train and a test set pair. The train set had 16 trials and the test set had 54. With trial length of 4 seconds, followed by rest of 2 seconds, the total length of the dataset is $1500 * (16s + 54s) * (4s + 2s) = 630000s$ or 175 hours of simulated EEG. A real recording session following the Graz BCI paradigm [41] would be much longer due to additional time needed for setup and dismantling of the electrode montage (from 7 to 20 minutes of setup per session in the study of Nijboer et al. [11]), possible initial burn-in period (e.g. 30sec per recording) and a few seconds overhead for cue on-

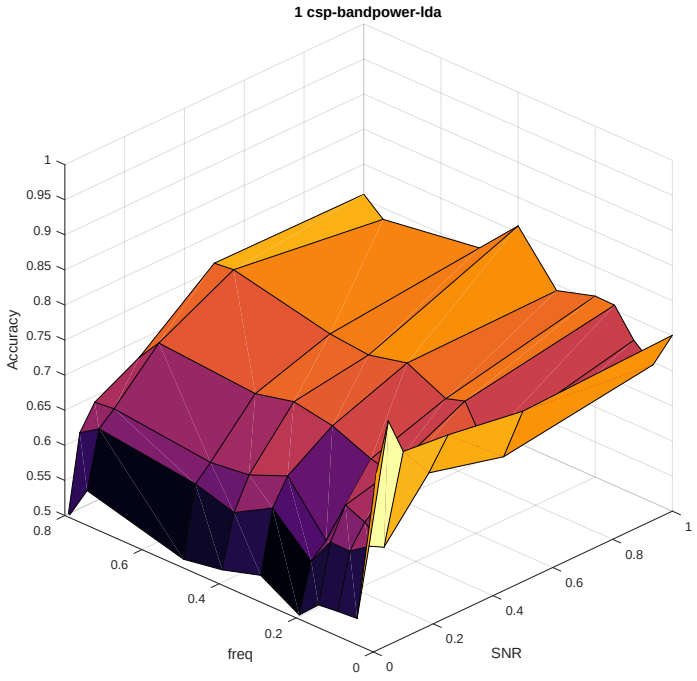


Figure 8: Effect of eyeblink frequency to simulated CSP/Bandpower/LDA two-class motor imagery classification. Each point is an average accuracy of 25 simulated experiments with resampled train and test sets. Low SNR indicates the eyeblink effect is stronger compared to the other signal components. Notice the apriori unintuitive result: having no eyeblinks is optimal for the accuracy, but many eyeblinks is generally better than having only a few of them. This may be due to the statistical pipeline possibly being able to take eyeblinks somewhat into account in the modelling when they are no longer rare. If eyeblinks are present, the SNR appears more important to the accuracy than the blink frequency. Approximately 175 hours of simulated BCI data was used to compute the plot.

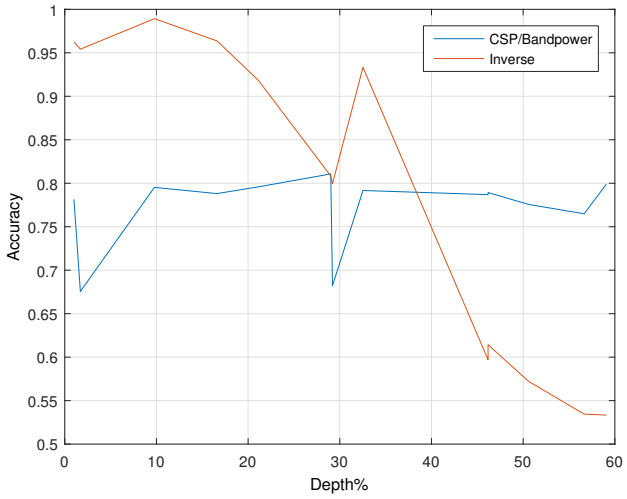


Figure 9: The effect of cortical depth of the signal generators on two signal processing pipelines. The decrease in accuracy for the inverse model based pipeline is due to the source going farther away from the fixed ROI specified for the the pipeline. For details, see text.

set/offset per trial. In contrast, the simulated experiment can be run on a normal desktop PC in one night.

Effect of location of the discriminable sources. Figure 9 shows another simulated experiment where we compare the CSP/Bandpower pipeline to a method based on inverse models, inspired by Cincotti & al. [49]. The first pipeline optimizes a statistical spatial filter, whereas the second model reconstructs the sources in the cortical volume before classification. The source reconstructing pipeline also removes source dipoles from consideration which are not inside a hand-specified region of interest (ROI) roughly covering the motor cortices. The first pipeline was illustrated in Figure 6, and the second is included in the framework distribution. The data generation is the same as before, using 16 training trials of 4 seconds each and 54 test trials. We disabled the eyeblink artifacts. Instead, we change the location of the sources that generate the discriminable signal. We move the two cortical dipoles downwards along the cortex, starting from their usual location under the electrodes C3 and C4 and then move the sources lower until the moved distance is approximately 60% of the diameter of the cortex model. It can be noted that the inverse pipeline performs well compared to the CSP/Bandpower, but its accuracy drops significantly when the sources eventually go out of the cortical ROIs specified for the inverse approach. However, it is of interest that this accuracy drop is not a on-off phenomenon.

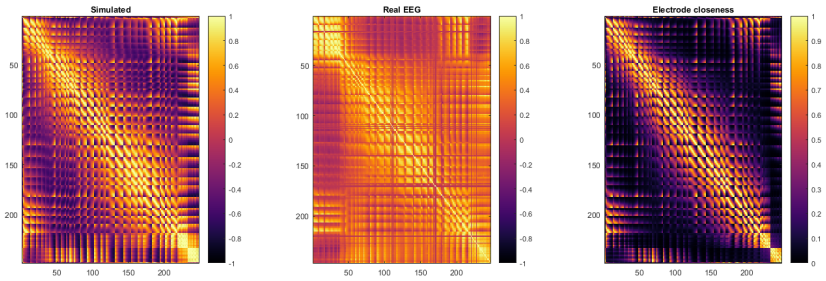


Figure 10: Correlation matrices of a specific kind of generated data (left) and real EEG data (middle). Right, a matrix showing pairwise electrode closeness measures. For details, see text.

This is due to a phenomenon called *source leakage* (e.g. [3]): the imprecision of the source reconstruction spreads the source activities to nearby sources when the sources are reconstructed. Thus, when the volumetric sources are reconstructed in the method, some relevant information leaks to the ROI even if the source is not originally inside it. This leakage is then caught by the statistical feature selection and classification techniques of the method, suggesting that the used BCI method can benefit from source leakage. As the source leakage is usually considered a harmful phenomenon in source reconstruction, this appears as another interesting finding that simulated studies can provide for further consideration. Finally, the plot shows an accuracy anomaly near 30% depth. As both pipelines are affected, this suggests a sudden change in how the leadfield projects the sources to the surface. Note that since we used a constrained orientation leadfield, the dipole traversal along the cortex changes the directions the source dipoles radiate to according to the folding of the cortical model.

Do electrode correlations require source correlations? In addition to providing quantitative results such as accuracies, the framework can be used to obtain qualitative insights about the model system. For example, using the framework to visualize the correlations of the generated data as in Figure 10 illustrates that the forward transform can introduce dependencies to the EEG. In the data, using the same head model as before, we specified all the sources in the volume to generate mutually independent pink noise. In the volume these sources were then approximately uncorrelated (not shown). However, after projection of the sources to the scalp EEG by the forward model, the surface data exhibits the typical EEG-like correlation patterns that in this case must emerge from the modeled volume conduction. For comparison, we also recorded a real dataset with an high-grade EGI system[4]

[4]256-channel EGI (Electrical Geodesics, USA) cap with EGI NetAmps 300 amplifier.

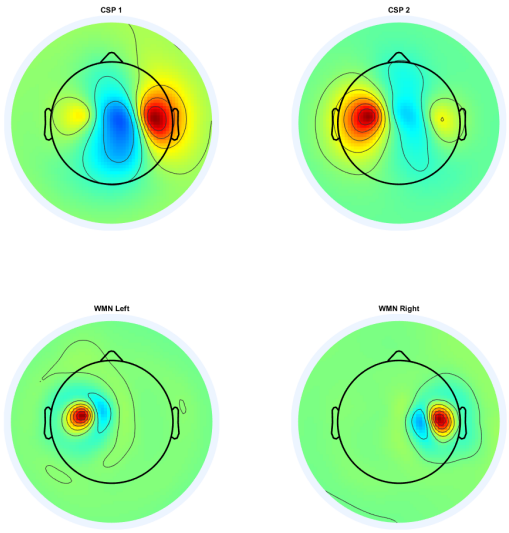


Figure 11: Spatial filters. Top, CSP filters obtained using simulated data. Bottow, filters from the WMN algorithm corresponding to reconstruction of the true sources. CSP uses only EEG data to build its model, whereas WMN relies on physiological information only and does not use data.

that the electrode set of the head model was modeled after. Note that the leadfield we used was not estimated from the same subject as the EEG was recorded from. Nevertheless, the correlation structure exhibits similar patterns both in simulated and real EEG. The correlations of real EEG data are stronger than those of the simulated data, suggesting that real brain activity has more dependencies in the volume than the independent noise we used. In both cases the correlations are qualitatively related to the electrode distance, as the image on the right in Figure 10 suggests. The image simply plots a Gaussian weighted distance between each electrode pair in the used electrode set.

The resemblance of CSP and WMN inverse model. Finally, for Figure 11, we generated one session of 2-class motor imagery data originating from 2 dipoles located on the left and right side motor cortexes and mixed these sources with pink volumetric noise as before, but with no artifacts. The figure illustrates the two most important CSP filters obtained with the data, as well as the corresponding projections obtained from the Weighted Minimum Norm estimate (WMN). The latter approach attempts to reconstruct the generating sources, using the generating leadfield and knowledge which sources generated the data, but not the data itself. The left and right are swapped in CSP as the algorithm does not order the filters in any manner. The integration performed by the WMN approach appears to focus tightly on the electrode locations close to the sources, whereas the CSP solution is less sparse and more spread across the electrodes. The CSP solution becomes less defined if eyeblink artifacts are present in the data, when the amount of training data is reduced, or the signal component is made weaker (not shown) but the WMN model is not affected by these changes as it does not use the data. With real data it would be more difficult to investigate the CSP reactions to different conditions, as the ground truth would not be available and some parameters such as the strength of Event-Related Desynchronization (ERD) or position of the sources would be difficult to modulate by a subject.

9 Discussion and future work

The proposed framework is not intended as a realistic brain simulator (a massive undertaking, see e.g. [8]). Even if the used models are made to match the current knowledge, some properties of the real brain ale likely to be presently unknown. The simulated data can only exhibit such behavior that can result from the models that are used. If some behavior of the real brain is outside the scope of these models, it cannot be discovered from the simulated data alone. On the contrary, analysis of real data may be able

to discover and use data characteristics which are outside the scope of the current generative models. Due to this, great care should be taken when drawing inferences regarding real BCI from simulated EEG. Instead of being a basis for such inferences, the framework should be used to gain insight about the interplay of the models being studied. Once some interesting hypotheses have been formulated, they can be used to guide the design of human experiments.

We stress that the user should apply our framework with consideration. For example, if a large set of different signal processing parameters are studied, it may be necessary to detect which differences are statistically significant and correct for multiple comparisons, for example using the functions from the Matlab's statistics toolbox. Even so, significant differences on simulated data do not guarantee the same differences on real data, and any findings should be taken only as hypotheses for additional study. Further, the data generation should be configured reasonably. As an example of the contrary, a long recording session of stationary EEG may be very difficult to obtain with a human subject, whereas for the simulator it is effortless. On the other hand, phenomena such as eyeblink frequency and modulating the strength of the Event-Related Desynchronization (ERD) in motor imagery may very well correspond to real phenomena.

In the future, the framework could be extended to many directions. One possibility involves adding connectivity modelling of brain areas over time. Another direction would involve implementing lower-level models of neuronal assemblies and even spiking neurons and model how their activities are aggregated into dipolar volume currents. The framework could also be extended to generate and test multiuser datasets for machine learning techniques that benefit from big data.

10 Conclusion

We have proposed a framework for studying BCI signal processing through simulation and have described the main aspects of its design. To summarize, the framework can generate and test multiclass EEG data and classification pipelines according to specifications given by the user. By its modular design, the framework allows mixing and matching of different kinds of signal generators, head models, BCI timelines, signal positioning, as well as noise and artifact generators. Similar mixing of parts is possible for the signal processing pipelines. For convenience, the framework can automatize the generation and testing of different parameter combinations of interest.

In the scope of the paper, we have illustrated that already simple simu-

lated experiments can reveal interesting and apriori counterintuitive hypotheses for further consideration. As the proposed framework is open source and available free of charge, we hope that the community will find the proposed system useful in illustrating BCI systems to students, in debugging existing BCI systems with artificial data, and most of all for studying a variety of questions that remain in BCI signal processing.

Acknowledgement

We thank Axelle Pillain, Lyes Rahmouni and John-Erick Guzmann for contributions to the framework. We also thank Camille Jeunet and Benoit le Gouis for comments on an earlier version of this manuscript. This work was funded by the Labex CominLabs project SABRE.

References

[1] B. Allison and C. Neuper. "Could Anyone Use a BCI?" In: *Brain-Computer Interfaces*. Ed. by D. S. Tan and A. Nijholt. Human-Computer Interaction Series. Springer, 2010, pp. 35–54. ISBN: 978-1-84996-271-1.

[2] L. Nicolas-Alonso, L. Fernando, and J. Gomez-Gil. "Brain Computer Interfaces, a Review." In: *Sensors (Basel)* 12.2 (2012), pp. 1211–1279.

[3] M. Clerc, L. Bougrain, and F. L. (eds). *Brain-Computer Interfaces*. Vol. 1-2. ISTE-Wiley, 2016.

[4] S. Baillet, J. C. Mosher, and R. Leahy. "Electromagnetic brain mapping". In: *IEEE Signal Processing Magazine* 18 (6 2001), pp. 14–30.

[5] J. T. Lindgren. "As above, so below? Towards understanding inverse models in BCI". In: *Journal of Neural Engineering* 15.1 (2017).

[6] W. J. Kaufmann and L. L. Smarr. *Supercomputing and the Transformation of Science*. New York, NY, USA: W. H. Freeman & Co., 1992. ISBN: 0716750384.

[7] D. Mourtzis, M. Doukas, and D. Bernidaki. "Simulation in Manufacturing: Review and Challenges". In: *Procedia CIRP* 25 (2014), pp. 213–229. ISSN: 2212-8271.

[8] H. Markram et al. "Introducing the Human Brain Project". In: *Procedia Computer Science* 7 (2011). Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11), pp. 39–42. ISSN: 1877-0509.

[9] F. Lotte et al. “A review of classification algorithms for EEG-based brain-computer interfaces”. In: *Journal of Neural Engineering* 4.2 (2007).

[10] B. Graimann, B. Allison, and G. Pfurtscheller. “Brain–Computer Interfaces: A Gentle Introduction”. In: *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*. Ed. by B. Graimann, G. Pfurtscheller, and B. Allison. Springer Berlin Heidelberg, 2010. Chap. 1, pp. 1–27.

[11] F. Nijboer et al. “Usability of three electroencephalogram headsets for brain-computer interfaces: a within subject comparison”. In: *Interacting with computers* 27.5 (2015). Ed. by H. Gamboa et al., pp. 500–511.

[12] G. Pfurtscheller et al. “EEG-based discrimination between imagination of right and left hand movement”. In: *Electroencephalography and Clinical Neurophysiology* 103.6 (1997), pp. 642–651.

[13] M. Grosse-Wentrup et al. “Beamforming in noninvasive brain-computer interfaces”. In: *IEEE Transactions on Biomedical Engineering* 56.4 (2009), pp. 1209–1219.

[14] M. Besserve, J. Martinerie, and L. Garnero. “Improving quantification of functional networks with EEG inverse problem: Evidence from a decoding point of view”. In: *NeuroImage* 55 (2011), pp. 1536–1547.

[15] P. L. Nunez and R. Srinivasan. *Electric Fields of the Brain, 2nd edition*. Oxford University Press, 2006.

[16] L. H. Zetterberg and K. Ahlin. “Analogue simulator of e.e.g. signals based on spectral components”. In: *Medical and biological engineering* 13.2 (Mar. 1975), pp. 272–278.

[17] A. Isaksson and A. Wennberg. “An EEG simulator—a means of objective clinical interpretation of EEG”. In: *Electroencephalography and Clinical Neurophysiology* 39.4 (1975), pp. 313–320.

[18] B. Kemp and F. H. L. da Silva. “Model-based analysis of neurophysiological signals”. In: *Digital Biosignal Proessing*. Ed. by R. Weitkunat. Elsevier, 1991, pp. 129–155.

[19] B. Kemp et al. “Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG”. In: *IEEE Transactions on Biomedical Engineering* 47.9 (Sept. 2000), pp. 1185–1194.

[20] D. Gutierrez, A. Nehorai, and C. H. Muravchik. “Estimating brain conductivities and dipole source signals with EEG arrays”. In: *IEEE Transactions on Biomedical Engineering* 51.12 (Dec. 2004).

[21] J. Yao and J. P. Dewald. "Evaluation of different cortical source localization methods using simulated and experimental EEG data". In: *NeuroImage* 25.2 (2005), pp. 369–382.

[22] A. Bradley et al. "Evaluation of Electroencephalography Source Localization Algorithms with Multiple Cortical Sources". In: *PLOS ONE* 11.1 (Jan. 2016), pp. 1–14.

[23] D. A. Bridwell et al. "Spatiospectral Decomposition of Multi-subject EEG: Evaluating Blind Source Separation Algorithms on Real and Realistic Simulated Data". In: *Brain Topography* (2016).

[24] S. Haufe et al. "A critical assessment of connectivity measures for EEG data: a simulation study". In: *Neuroimage* 64 (2013), pp. 120–133.

[25] F. Tadel et al. "Brainstorm: A User-Friendly Application for MEG/EEG Analysis". In: *Computational Intelligence and Neuroscience* 2011 (2011).

[26] R. Oostenveld et al. "FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data". In: *Computational Intelligence and Neuroscience* (2011).

[27] A. Delorme et al. "EEGLAB, SIFT, NFT, BCILAB, and ERICA: New Tools for Advanced EEG Processing". In: *Computational Intelligence and Neuroscience,* (2011).

[28] J. d. R. Millán et al. "Combining Brain–Computer Interfaces and Assistive Technologies: State-of-the-Art and Challenges". In: *Frontiers in Neuroscience* 4 (2010), p. 161.

[29] D. Boland et al. "Using Simulated Input into Brain-Computer Interfaces for User-Centred Design". In: *International Journal of Bioelectromagnetism* 13.2 (2011), pp. 86–87.

[30] D. A. Rohani et al. "BCI using imaginary movements: The simulator". In: *Computer Methods and Programs in Biomedicine* 111.2 (2013), pp. 300–307.

[31] C. Brunner et al. "Improved signal processing approaches in an offline simulation of a hybrid brain–computer interface". In: *Journal of Neuroscience Methods* 188.1 (2010), pp. 165–173. ISSN: 0165-0270.

[32] F. Lotte. "Generating Artificial EEG Signals To Reduce BCI Calibration Time". In: *5th International Brain-Computer Interface Workshop, Graz*. 2011, pp. 176–179.

[33] M. Tangermann et al. "Review of the BCI Competition IV." In: *Frontiers in neuroscience* 6 (2012).

[34] C. Brunner et al. “BCI software platforms”. In: *Towards Practical Brain-Computer Interfaces*. Springer Berlin Heidelberg, 2013, pp. 303–331.

[35] J. Lindgren and A. Lecuyer. “OpenViBE and Other BCI Software Platforms”. In: *Brain–Computer Interfaces 2: Technology and Applications*. John Wiley & Sons, 2016.

[36] Y. Renard et al. “OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments”. In: *Presence : teleoperators and virtual environments* 19.1 (2010).

[37] J. Mellinger and G. Schalk. “BCI2000: A General-Purpose Software Platform for BCI Research”. In: *Toward Brain-Computer Interfacing*. Ed. by G. Dornhege and J. M. et al. MIT Press, 2007, pp. 372–381.

[38] R. Grech et al. “Review on solving the inverse problem in EEG source analysis”. In: *Journal of NeuroEngineering and Rehabilitation* 5.25 (2008).

[39] A. Gramfort et al. “OpenMEEG: opensource software for quasistatic bioelectromagnetics”. In: *BioMedical Engineering OnLine* 9.45 (2010).

[40] F. B. Vialatte et al. “Steady-state visually evoked potentials: Focus on essential paradigms and future perspectives”. In: *Progress in Neurobiology* 90.4 (2010), pp. 418–438.

[41] G. Pfurtscheller and C. Neuper. “Motor imagery and direct brain-computer communication”. In: *Proceedings of the IEEE* 89.7 (2001), pp. 1123–1134.

[42] R. Fazel-Rezai et al. “P300 brain computer interface: current challenges and emerging trends”. In: *Frontiers in Neuroengineering* 5.14 (2012).

[43] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. “Designing optimal spatial filters for single-trial EEG classification in a movement task”. In: *Clinical Neurophysiology* 110.5 (1999), pp. 787–798.

[44] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, 2nd edition*. Springer, 2008.

[45] R.-E. Fan et al. “LIBLINEAR: A library for large linear classification”. In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874.

[46] B. J. Edelman, B. Baxter, and B. He. “EEG Source Imaging Enhances the Decoding of Complex Right-Hand Motor Imagery Tasks”. In: *IEEE Transactions on Biomedical Engineering* 63 (1 2016), pp. 4–14.

[47] R. Pascual-Marqui. “Standardized low-resolution brain electromagnetic tomography (sLORETA): technical details”. In: *Methods Find Exp Clin Pharmacol* 24 (2002), pp. 5–12.

[48] A. J. Bell and T. J. Sejnowski. “An information maximisation approach to blind separation and blind deconvolution”. In: *Neural Computation* 7.6 (1995), pp. 1129–1159.

[49] F. Cincotti et al. “High-Resolution EEG Techniques for Brain-Computer Interface Applications”. In: *Journal of Neuroscience Methods* 167.1 (2008).

[50] R. Oostenveld et al. “FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data”. In: *Computational intelligence and neuroscience* 2011 (2011), p. 1.

[51] T. F. Oostendorp, J. Delbeke, and D. F. Stegeman. “The conductivity of the human skull: results of in vivo and in vitro measurements”. In: *IEEE transactions on biomedical engineering* 47.11 (2000), pp. 1487–1492.