OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments

Yann Renard[1,*], Fabien Lotte[1], Guillaume Gibert[2],

Marco Congedo[3], Emmanuel Maby[2], Vincent Delannoy[1],

Olivier Bertrand[2], Anatole Lécuyer[1,*]

[1]INRIA Rennes, FRANCE

[2]INSERM U821, Lyon, FRANCE

[3]INPG Gipsa-Lab, Grenoble, FRANCE

*Corresponding authors:

Yann Renard	Anatole Lécuyer
Bunraku Team	Bunraku Team
INRIA Rennes - Bretagne Atlantique	INRIA Rennes - Bretagne Atlantique
Campus universitaire de Beaulieu	Campus universitaire de Beaulieu
35042 Rennes Cedex	35042 Rennes Cedex
FRANCE	FRANCE
yann.renard@irisa.fr	anatole.lecuyer@irisa.fr

Abstract

This paper describes the OpenViBE software platform which enables to design, test and use Brain-Computer Interfaces. Brain-Computer Interfaces (BCI) are communication systems that enable users to send commands to computers only by means of brain activity. BCI are gaining interest among the Virtual Reality (VR) community since they have appeared as promising interaction devices for Virtual Environments (VE). The key features of the platform are 1) a high modularity, 2) embedded tools for visualization and feedback based on VR and 3D displays, 3) BCI design made available to non-programmers thanks to visual programming and 4) various tools offered to the different types of users. The platform features are illustrated in this paper with two entertaining VR applications based on a BCI. In the first one, users can move a virtual ball by imagining hand movements, while in the second one, they can control a virtual spaceship using real or imagined foot movements. Online experiments with these applications together with the evaluation of the platform computational performances showed its suitability for the design of VR applications controlled with a BCI. OpenViBE is a free software distributed under an open-source license.

1 Introduction

One of the keys to a great immersion feeling with Virtual Reality (VR) is the ease of interaction with Virtual Environments (VE). Recently, a new method has emerged: interacting through cerebral activity, using a Brain-Computer Interface (BCI) (Leeb et al., 2007, 2006; Lécuyer et al., 2008). Such an interface is a communication system that enables a user to send commands to a computer by means of variations of brain activity, which is in turn measured and processed by the system (Wolpaw, Birbaumer, McFarland, Pfurtscheller, & Vaughan, 2002). BCI are currently following the path drawn by haptic devices a few years ago (Burdea, 1996) by providing a completely new way of conceiving interaction with computers and electronic devices through the "interaction by thought" concept. The BCI technology is rapidly improving, and several interesting applications using BCI have already been developed for navigating or interacting with virtual environments (Leeb et al., 2007; Lécuyer et al., 2008; Friedman et al., 2007), or for videogames (Krepki, Blankertz, Curio, & Müller, 2007; Nijholt, 2009).

However, designing BCI-based interaction devices requires expertise in a broad range of domains, ranging from neurophysiology, signal processing and interaction, to computer graphics or computer programming which represents a challenging multidisciplinary task. A general purpose software platform that provides the necessary functionalities to easily design BCI and connect them with VR would foster the research in the domain and democratize the use of BCI in real and virtual environments.

In this paper, we present the OpenViBE platform, a novel, free and open source platform to design and tune BCI systems and connect them with real and virtual environments. This paper is organized as follows: section 2 proposes a short state-of-the-art of existing BCI platforms while section 3 describes the features of our platform. Section 4 presents the range of users our system targets. Sections 5 and 6 detail respectively the design of a BCI with OpenViBE and the tools we provide. Section 7 is dedicated to the connection with VR and section 8 details the platform internals. Finally, some examples of BCI implementations, performances and current state of the platform are presented respectively in sections 9, 10 and 11. The paper ends with a general conclusion.

2 Related work: existing BCI software

Several software for offline and online analysis of EEG and biomedical signals are available. They are briefly reviewed in (Schlögl, Brunner, Scherer, & Glatz, 2007). However, these software do not include all the necessary functionalities for designing a BCI.

In the freeware community, only three software enclose the necessary functionalities for real-time BCI designs: BioSig (Schlögl et al., 2007) (thanks to the "rtsBCI" package), BCI2000 (Mellinger & Schalk, 2007) and BCI++(Maggi, Parini, Perego, & Andreoni, 2008).

BioSig is an open-source software library for biomedical signal processing and more specifically

for BCI research (Schlögl et al., 2007). It is a toolbox for Octave and Matlab which offers several data management modules, data import and export, artifact processing, quality control, feature extraction algorithms, classification methods, etc. It also offers rapid prototyping of online and real-time BCI with the "rtsBCI" package using Matlab/Simulink.

BCI2000 is a general-purpose system for BCI research (Mellinger & Schalk, 2007). This software is not open-source but its sources and executables are available for free for non-profit research and educational purposes. BCI2000 is a C++ software that proposes to build an online and real-time BCI by assembling four modules: the source module, for data acquisition and storage ; the signal processing module, that comprises the preprocessing, feature extraction and classification of brain activity ; the user application module, with which the user interacts ; and finally, the operator interface for data visualization and system configuration. Interestingly, BCI2000 also provides tools for offline analysis of data within the "MARIO" software.

Recently, another BCI software platform has been proposed: BCI++ (Maggi et al., 2008). This software is a C/C++ framework for designing BCI systems and experiments. BCI++ also includes some 2D/3D features for BCI feedback. However, it should be mentioned that this platform is not completely open-source.

A comparison of these software with OpenViBE is provided in section 3.1.

3 OpenViBE features

OpenViBE is a free and open-source software platform for the design, test and use of Brain-Computer Interfaces. The platform consists of a set of software modules that can be easily and efficiently integrated to design BCI for both real and VR applications. Key features of the platform are:

Modularity and reusability. Our platform is a set of software modules devoted to the acquisition, pre-processing, processing and visualization of cerebral data, as well as to the interaction with VR displays. OpenViBE being a general purpose software implies that users are able to easily add new software modules in order to fit their needs. This is ensured thanks to the box concept, an elementary component in charge of a fraction of the whole processing pipeline, that allows to develop reusable components, reduces development time and helps to quickly extend functionalities.

Different types of users. OpenViBE is designed for different types of users: VR developers, clinicians, BCI researchers, etc. Their various needs are addressed and different tools are proposed for each of them, depending on their programming skills and their knowledge in brain processes.

Portability. The platform operates independently from the different software targets and hardware devices. It includes an abstract level of representation allowing to run with various acquisition machines, such as EEG or MEG. It can run on Windows and Linux operating systems

and also includes different data visualisation techniques. Finally, it is based on free and portable software (e.g., GTK+[1], IT++[2], GSL[3], VRPN[4], GCC[5]).

Connection with VR. Our software can be integrated with high-end VR applications. OpenViBE acts as an external peripheral to any kind of real and virtual environments. It also takes advantage of VR displays thanks to a light abstraction of a scenegraph management library, allowing to visualize cerebral activity in a legible way or to provide incentive training environments (e.g., for neurofeedback).

---

[1]The Gnome ToolKit is a highly usable, feature rich toolkit for creating graphical user interfaces which boasts cross platform compatibility and offers an easy to use API. More information can be found at http://www.gtk.org

[2]IT++ is a C++ library of mathematical, signal processing and communication routines. More information can be found at http://sourceforge.net/apps/wordpress/itpp

[3]The GNU Scientific Library is a numerical library for C and C++ programmers. More information can be found at http://www.gnu.org/software/gsl

[4]The Virtual-Reality Peripheral Network is a set of classes within a library designed to implement an interface between application programs and the set of physical devices used in a virtual-reality system. More information can be found at http://www.cs.unc.edu/Research/vrpn

[5]The GNU Compiler Collection is a compiler which supports a wide range of architectures. More informations can be found at http://gcc.gnu.org

3.1 Comparison with other BCI platforms

In comparison to other BCI software, the OpenViBE platform appears as highly modular. It addresses the needs of different types of users (should they be programmers or non-programmers) and proposes a user-friendly graphical language which allows non-programmers to design a BCI without writing a single line of code. In contrast, all other BCI platforms require some degree of programming skills to design a new real-time BCI from scratch. Furthermore, their modularity is coarser (except for BioSig), hence restricting the range of possible designs.

OpenViBE is also portable, independent of the hardware or software and is entirely based on free and open-source software. In comparison, among other real-time BCI platforms, only BioSig is fully open-source but the "rtsBCI" package needed for online and real-time BCI requires Matlab/Simulink which a non-free and proprietary software.

OpenViBE proposes to generate online scenarios (step 3 in Figure 1) automatically from offline analysis (step 2 in Figure 1). Finally, in contrast with other platforms, OpenViBE is well suited for VR applications as it provides several embedded tools to design innovative VR displays and feedback as well as to perform 3D visualization of brain activity in real-time. Furthermore, OpenViBE can also be used as a device for any VR application.

4 Different types of users

OpenViBE has been designed for four types of users. On the first hand, the developer and the application developer are both programmers, on the other hand the author and the operator do not need any programming skills.

The developer (programmer) has the possibility to add new functionalities and test his own pieces of software in OpenViBE. To that end, OpenViBE is delivered with a complete Software Development Kit (SDK). This SDK provides access to functionalities at different levels depending on the task to realize. There are two main categories of developers. First, the kernel developers who enhance and modify kernel functionalities (see section 8.2). Second, plugin developers who create new additional modules (see section 8.3).

The application developer (programmer) uses the SDK to create standalone applications, using OpenViBE as a library. Such applications range from new tools such as the visual scenario editor described in section 6, to external VR applications that the BCI user can interact with. Such VR applications are presented in section 7.

The author (non-programmer) uses the visual scenario editor (see Figure 2) to arrange existing boxes to form a scenario. He configures these boxes and the scenario in order to produce a complete, ready-to-use BCI system. The author is aware of the internals of our platform as well as of BCI systems and is familiar with basic signal processing. He is also aware of the interaction paradigm to use. However, he does not need strong computer programming skills because he uses

dedicated tools to perform his tasks (see section 6).

The operator (non-programmer) generally would be a clinician or a practitioner (he is not a computer expert nor an OpenViBE expert). He is in charge of using and running the pre-built scenarios of the author. He then simply runs the scenario. He is aware of how the BCI system should and can work, and monitors the execution of the BCI system thanks to dedicated visualization components. He has understanding of neurophysiological signals and can help the BCI user to improve his control over the BCI system.

Finaly, another role should be considered: the BCI user. The BCI user generally wears the brain activity acquisition hardware (e.g., an EEG cap) and interacts with an application by means of his mental activity. The application could be for instance, a neurofeedback training program, a videogame in virtual reality, a remote operation in augmented reality, etc. While he does not directly use the OpenViBE platform, he implicitly takes advantage of its features.

5 How to design a BCI with OpenViBE?

Designing and operating an online BCI with our software follows a rather universal way of doing so (Wolpaw et al., 2002). Three distinct steps are required (see Figure 1). In the first step, a training dataset must be recorded for a given subject, while he performs specific mental tasks. The second step consists in an offline analysis of these records with the goal of finding the best

calibration parameters (e.g. optimal features, relevant channels, etc.) for this subject. The last step consists in using the BCI online in a closed loop process. Optionally, iterations can be done on data acquisition and offline training in order to refine the parameters.

The online loop (third step) is common to any BCI and it is composed of six phases: brain activity measurements, preprocessing, feature extraction, classification, translation into a command and feedback (see Figure 1).

Figure 1 here.

Brain activity measurements: This step consists in measuring the brain activity of the BCI user. To date, about half a dozen different kinds of brain signals have been identified as suitable for a BCI, i.e., easily observable and controllable (Wolpaw et al., 2002). Measuring the brain activity for a BCI system is mainly performed using electroencephalography (EEG) since it is a cost-effective and non-invasive method which provides a high temporal resolution (Wolpaw et al., 2002). Our software already supports various EEG acquisition devices but also supports a magnetoencephalography (MEG) machine (see Section 11 for a list of supported devices).

Preprocessing: The preprocessing step aims at denoising the acquired signals and/or at enhancing a specific brain signal (Bashashati, Fatourechi, Ward, & Birch, 2007). For example, our software proposes different kinds of preprocessing algorithms such as temporal filters and

spatial filters (independent component analysis, surface Laplacian, etc.).

Feature Extraction: Once signals have been preprocessed, features can be extracted. These features consist in a few values that describe the relevant information embedded in the signals (Bashashati et al., 2007) such as the power of the signals in specific frequency bands (Pfurtscheller & Neuper, 2001). These features are then gathered into a vector called "feature vector". Examples of features available in OpenViBE include band power features or power spectral densities.

Classification: The feature vector is fed into an algorithm known as "classifier". A classifier assigns a class to each feature vector, this class being an identifier of the brain signal that has been recognized. In general, the classifier is trained beforehand using a set of feature vectors from each class. An example of classifier used for BCI would be the Linear Discriminant Analysis (Lotte, Congedo, Lécuyer, Lamarche, & Arnaldi, 2007). It should be noted that, due to the high variability and noisiness of EEG signals, classification rates of 100 % are very rarely attained, even for a BCI using two mental states. OpenViBE proposes several classifiers such as Linear Discriminant Analysis (Lotte, Congedo, et al., 2007) or Fuzzy Inference Systems (Lotte, Lécuyer, Lamarche, & Arnaldi, 2007).

Translation into a command: Once the class of the signal has been identified, it can be associated to a command which is sent to a computer in order to control, for instance, a robot (Millán, 2008) or a prosthesis (Wolpaw et al., 2002). The number of possible commands in

current EEG-based BCI systems typically varies between 1 and 4.

Feedback: Finally, feedback should be provided to the user so that he can determine whether he correctly performed the brain signal. This is an important step as it helps the user to control his brain activity (Lotte, Renard, & Lécuyer, 2008; Neuper, Scherer, Wriessnegger, & Pfurtscheller, 2009). Feedback can be simple visual or audio cues, e.g., gauges. To this aim, our software proposes classical raw signal, spectra, time/frequency visualisation modules. Alternatively, more advanced feedback can be provided such as the modification of a virtual environment (Leeb et al., 2007) to which OpenViBE send commands.

6 Tools

Our system includes a number of usefull tools for its various users: the acquisition server, the designer, 2D visualization tools and sample scenarios of BCI or neurofeedback.

The acquisition server provides a generic interface to various kinds of acquisition machines, e.g., EEG or MEG systems. Such an abstraction allows the author to create hardware independent scenarios, thanks to the use of a generic acquisition box. This box receives the data via the network from the acquisition server, which is actually *connected* to the hardware and transforms these data in a generic way. The way the acquisition server gets *connected* to the device mostly depends on the hardware manufacturer's way to access his device. Some devices will be shipped

with a specific Software Development Kit, some others will propose a communication protocol over a network/serial/USB connection. Finally, some devices will need a proprietary acquisition software that delivers the measures to our own acquisition server.

The designer is mainly dedicated to the author and enables him to build complete scenarios based on existing software modules using a dedicated graphical language and a simple Graphical User Interface (GUI) as shown in Figure 2. The author has access to a list of existing modules in a panel, and can drag and drop them in the scenario window. Each module appears as a rectangular box with inputs (on top) and outputs (at the bottom). Double clicking on a box displays its configuration panel. Boxes are manually connectable through their inputs and outputs. The designer also allows the author to configure the arrangement of visualization windows (i.e., visualization modules included in the scenario). An embedded player engine allows the author to test and debug his scenario in real time. In doing so, the author can receive a continuous feedback on boxes status and their processing times. Such a feedback may be useful to balance the computational load.

Figure 2 here.

The 2D visualization features of the platform are available as specific boxes and include brain activity related visualizations. These boxes can access all the platform functionalities, and

particularly the whole stream content for the connected inputs. Most 2D visualization boxes display input data in a widget and do not produce output. Our system offers a wide range of visualization paradigms such as raw signal display, gauges, power spectrum, time-frequency map and 2D topography in which EEG activity is projected on the scalp surface in two dimensions (see figure 3). OpenViBE also provides a visualization tool which displays instructions to a user according to the protocol of the famous Graz motor imagery-based BCI (Pfurtscheller & Neuper, 2001).

Figure 3 here.

Existing and pre-configured ready-to-use scenarios are proposed to assist the author. As the creation of new scenarios has been made fast and easy, the number of available scenarios is expected to rapidly increase. Currently, five complete scenarios are available:

- Hand motor imagery based BCI: this scenario allows to use OpenViBE as an interaction peripheral using imagined movements of the left and right hand. This scenario is inspired from the well-known Graz-BCI of the Graz University (Pfurtscheller & Neuper, 2001) (see section 9).
- Self-paced BCI based on foot movements: this scenario represents a BCI based on real or imagined foot movements that can be used in a self-paced way. This means the subject can

interact with the application at any time, contrary to most existing BCI (see section 9).

- Neurofeedback: this scenario shows the power of the brain activity in a specific frequency band, and helps a subject in the task of self-training to control that power.

- Real-time visualization of brain activity in 2D/3D: this scenario enables the user to visualize his own brain activity evolving in real-time on a 2D or 3D head model. This scenario can be used together with inverse solution methods (Baillet, Mosher, & Leahy, 2001), in order to visualize the brain activity in the whole brain volume, not only on the scalp surface, as in (Arrouët et al., 2005) (see Figure 6).

- P300-speller: this scenario implements the famous P300-speller (Farwell & Donchin, 1988; Donchin, Spencer, & Wijesinghe, 2000), an application which enables a user to spell letters by using only his brain activity, and more precisely the event related potential known as the P300 (Wolpaw et al., 2002) (see Figure 4 and Figure 6). It should be noted that OpenViBE can also be used to design other P300-based BCI, and not only the P300-speller. Interested readers may refer to (Sauvan, Lécuyer, Lotte, & Casiez, 2009) for another example of a P300-based BCI designed with OpenViBE.

Figure 4 here.

Figure 5 here.

Figure 5 summarizes how users interact with software and hardware components. For example, the operator uses both the acquisition server and the designer; the brain activity acquisition device feeds the acquisition server and the analysis is performed on the computer hosting the designer. This results in a dedicated embedded visualization that monitors the user's brain activity.

7 Connection with Virtual Reality

The platform includes a number of embedded 3D visualization widgets and is able to interact with external VR applications thanks to standard communication protocols. However, OpenViBE does not aim at offering a complete set of scenegraph managing capabilities, nor at embedding a VR application builder. Consequently, OpenViBE users should be aware of what the platform is in charge of, and what is left to external applications communicating with OpenViBE.

OpenViBE as an interaction device for external Virtual Reality applications: our platform can first be used as an interaction peripheral for any general purpose application in real and virtual environments. As such, some of the data processed by the scenario need to be exposed to the outside world. There are two ways this goal can be achieved.

One way is to propose specific boxes which can expose parameters in a “considered as standard” way. For example, the platform includes a VRPN module (Taylor et al., 2001) that acts as a server and sends analogic and button values. This is a convenient way to interact with existing VR applications. The advantage stands in that VR application developers do not have to perform major modifications on their application to have it controlled by a BCI user. Examples of VR applications using OpenViBE and the VRPN plugin are given in section 9.

The other way is to build an application using the platform as a third-party peripheral management library. The developer has access to the whole exposed data and is able to process and display it in his own application.

OpenViBE for direct visualization and interaction with 3D models: In order to perform 3D visualization, with or without VR displays, the OpenViBE kernel hides a scenegraph manager and exposes a number of functionalities such as color, position, transparency and scale settings for 3D objects, as well as mesh management capabilities. This allows developers to easily and quickly develop 3D plugins using a simplified 3D Application Programming Interface (API). This API offers the required functionalities to load and dynamically modify a 3D scene based on the input data and allows direct visualization and interaction with 3D models.

OpenViBE for real-time visualization of brain activity: OpenViBE is also used to visualize brain activity in real-time or to get immmersive neurofeedback. In order to achieve this, the scenegraph manager is used by several visualisation widgets. Figure 6 shows two examples of

what our platform offers in terms of embedded 3D widgets for real-time visualization of brain activity: a 3D topographic map which displays the recorded potentials mapped onto a 3D head and a voxelized reconstruction of the inside brain activity, based on scalp measures.

Figure 6 here.

8 OpenViBE internals

This section describes the software architecture of the platform. In order to design an extensible software, we followed the approach of already existing and widely used VR software such as Virtools (*Virtools website*, 2007). In such software, the classical kernel and plugin architecture ensures maximum extensibility. A new plugin can be dynamically added and used by the kernel for the applications' benefit, without the need to rebuild the application or the kernel itself. Additionally, composing scenarios based on elementary components ensures maximum flexibility and reusability.

Therefore, each application of our platform relies on a common kernel which delegates tasks to a number of dedicated plug-ins as shown in Figure 7. Moreover, the kernel offers the concept of box, allowing the creation of powerful tools such as the designer authoring tool. Each of these

components are presented in the following sections.

Figure 7 here.

8.1 The box concept

The box is a key component of the platform. It consists of an elementary component in charge of a fraction of the whole processing pipeline. It exposes inputs and outputs to other boxes. Each box can be notified on clock ticks and upon input data arrival. The behavior of a box can be adapted to the needs of each algorithm (for instance, acquisition algorithms typically react to clock signals whereas processing algorithms typically react to input arrival). The characteristics and constraints that are common to all boxes include reasonable granularity to allow quick software components rearrangement. Newly developed boxes are immediately available to the user thanks to the plugin system (see section 8.3).

8.2 The kernel

The kernel provides global services to applications through several managers, each of them providing a set of specialized services.

For example, the **plug-in manager** makes the platform extensible. This manager is able to

dynamically load plug-in modules (e.g., .DLL files under Windows, or .so files under Linux) and collect extensions from them, such as scenario serializers, algorithms and boxes (see section 8.3). The plug-in system allows to quickly and efficiently expand functionalities. The communication interface between these extensions and the kernel itself is defined so that they can easily be shared, used and replaced when needed.

Another example is the **scenario manager** which helps creating and configuring scenarios. For instance, the manager can add or remove boxes, change their settings and connect them altogether. The scenario manager can handle multiple scenarios simultaneously. The designer authoring tool takes advantage of this in order to edit them in multiple tabs.

Finaly, the **visualization manager** is responsible for displaying 2D or 3D graphical information and setting their position and size in a window. Indeed, multiple visualization windows may be used. The windows arrangement in space is done by the visualization manager at editing time, thanks to the designer application, and saved to a file. Basic signal display windows are provided with a 2D rendering context (see Figure 3), while more advanced rendering is performed thanks to the encapsulated 3D library (see section 7).

Several other managers exist such as the **player manager** for an easy setup of a runtime session, the **configuration manager** for a convenient way to configure the whole platform with text files and the **type manager** which ensures coherency and possibly conversions of all data types (e.g. box settings or connectors). Interested readers will find more information about those managers

in the software documentation (INRIA, 2009).

8.3 Plug-ins

Our platform includes three different families of plug-in:

The driver plug-ins allow to add acquisition devices to the acquisition server. A driver basically reads the signal from the device through a specific Software Development Kit or a physical connection, and injects this signal into OpenViBE in a generic way. The rest of the processing pipeline is therefore independent of the acquisition hardware.

The algorithm plug-ins are a generic abstraction for any extension that could be added to the platform (e.g., add new feature extraction or signal processing methods). Algorithms are the developer's atomic objects. The developer may compose several algorithms in order to achieve a complex task. This kind of plugin allows to massively share and reuse software components, even in an offline context where time is handled at a different scale (e.g., EEG file reading or signal visualization widgets).

The box plug-ins are the software components each box relies on. Boxes are the author's atomic objects. The developer describes them in a simple structure that notably contains the box prototype (its name, input/output connectors and settings). The box is responsible for the actual processing, i.e., it reads from inputs, computes data to produce a result and writes to outputs. The box generally combines several algorithm plug-ins together to perform its processing. This

ensures fast development thanks to the re-usability of components.

Additionally, it should be stressed that a specific box is available to developers: a box that accepts Matlab code. This box aims at providing a tool to quickly develop and test some algorithms using the Matlab language. As soon as the prototype gets functional, it can be implemented in C++ for better performances.

9 Examples of implementation

In this section, we illustrate the capabilities of our software and the way to use it as an interaction device with two immersive applications: the “Handball” and the “Use-the-force” applications. The description of the “Handball” application includes a special emphasis on the way OpenViBE is used to design the BCI and its associated scenarios.

9.1 The “Handball” application

The Handball VR application is an immersive 3D game in which the user can control a virtual ball by using a BCI based on imagined hand movements. The game objective is to bring this ball into a goal cage. As such, this application enables to illustrate the use of OpenVIBE for the design of a very popular kind of BCI, namely a motor-imagery based BCI (Pfurtscheller & Neuper, 2001), and its use for interaction with a VR application. This section briefly describes the

BCI system and the VR game, then it details how OpenViBE is used in the implementation of this application and reports on an online experiment with real subjects.

9.1.1 BCI system

For this application, we used a motor-imagery based BCI which is inspired from the well known Graz BCI (Pfurtscheller & Neuper, 2001). Such BCI have been used successfully with several VR applications (Friedman et al., 2007; Leeb et al., 2006). With this system, the user has to perform imagined movements of his left or right hand to generate the brain signals expected by the BCI. It is established that performing an imagined hand movement triggers EEG power variations in the μ ($\simeq$ 8-13 Hz) and β ($\simeq$ 13-30 Hz) frequency bands, over the motor cortices (Pfurtscheller & Neuper, 2001). Consequently, to identify these specific variations, the BCI uses logarithmic Band Power (BP) for feature extraction. Such features are simply computed by band-pass filtering the signal in subject-specific frequency bands (roughly in the μ and β bands), squaring it, averaging it over a given time-window and computing its logarithm. Such features are extracted from the EEG channels located over the motor cortex. The generated feature vector is then passed to an efficient and widely used classifier, the Linear Discriminant Analysis (LDA), which will identify the signal class, i.e., “left” or “right”, depending on the hand chosen for the imagined movement. In order to improve the performance of the BCI, we also used temporal and spatial filtering as preprocessing (see section 9.1.3 for details).

9.1.2 Virtual reality game

The VE for this application was a virtual gymnasium equipped with a handball playing court and its two goal cages. The two goals were located on each side of the screen (see Figure 8); virtual ball was also located in this VE and could be controlled by the user as part of the game. Each time the BCI system recognized an imagined left hand movement in the brain activity, an event was sent to the VE, and the ball rolls towards the left goal. By symetry, the detection of an imagined right hand movement triggers the ball to roll towards the right goal. It should be noted that this application operated in a synchronous mode, which means the user could move the ball only during specific time periods, instructed by the system. As part of this game, the player's objective was to bring the ball into one of these two goals, as instructed by the application. More precisely, a game session was composed of 40 trials, among which 20 instructed the user to score in the left goal and 20 in the right goal. The order of the trials was randomized within a session. A trial was arranged as follows: at the beginning fo the trial (t=0s), the ball was located at the center of the playing ground, i.e., at the center of the screen. At t=2s, the ball color changed from red to green to indicate the user he should get ready to perform motor imagery to move the ball. At t=3.25s, a downward pointing arrow appears above one of the two goals to indicate the target goal. From t=4s to t=8s the user can move the ball continuously by using motor imagery and should try to reach the target goal. At the end of the trial, the ball automatically goes back to the screen center. The user scores a point if, at the end of the trial, the ball is closer to the target goal than to the

other. A trial is followed by a short rest period of random duration. It should be noted that the experimental paradigm used in this application is equivalent to that of the Graz BCI protocol (Pfurtscheller & Neuper, 2001).

Figure 8 here.

9.1.3 Implementation with OpenViBE

As mentioned previously, before using a BCI, an offline training phase is required in order to calibrate the system. This training phase needs a set of sample EEG signals. Consequently, the implementation of the BCI of this application is divided into four OpenViBE scenarios: three scenarios for the calibration of the BCI (acquisition of training data, selection of subject-specific frequency bands and classifier training) and one scenario for the online use of the BCI.

Step 1: Acquisition of training data. This phase aims at collecting training EEG data recorded while the subject performs imagined left or right hand movements. The scenario corresponding to this phase simply consists in the assembly of four boxes. The first one is a generic network acquisition box which acquires the recorded EEG signals. The second is a file writer box, which writes these EEG signals into a file using the GDF (General Data Format) format (Schlogl, 2006). The next box is a visualization box which to displays the instructions that the user will have to

follow. These instructions are: perform an imagined movement of the left or right hand, rest, etc. Finally, a stimulation box is used. This box generates events according to an XML file passed as parameter. These events are sent to the visualization box, which will display the corresponding instructions, and to the file writer box, which will store the events in order to know when the subject was asked to perform imagined movements. These events are generated according to the Graz BCI protocol (Pfurtscheller & Neuper, 2001).

Step 2: Offline training. This phase consists in determining the optimal BCI parameters for the subject, i.e., the optimal frequency bands for discriminating the two brain states using BP features, and the parameters of the LDA classifier. The optimal frequency bands are obtained using a statistical analysis on the training EEG signals, as in (Zhong, Lotte, Girolami, & Lécuyer, 2008; Lotte, Lécuyer, et al., 2007). The LDA classifier is then trained on the BP features extracted from the EEG training data. This offline training phase is decomposed into two scenarios: one for selecting the optimal frequency bands and one for training the classifier. For these two scenarios, three specific boxes are necessary: a GDF file reader, in charge of reading the data recorded during the previous phase, a statistical analysis box that will estimate the best frequency bands in which to extract the BP features, and a LDA training box to train the classifier on these features. All obtained parameters are saved for further use, i.e., during the online phase. It is worth noting that once the training is achieved, two pieces of scenario are generated: one contains the assembly of boxes that are necessary to extract BP features in the selected frequency bands and another

contains the trained LDA classifier.

Step 3: Online use of the BCI. The last phase is the online use of the BCI. The OpenViBE scenario corresponding to this phase is displayed in Figure 9.

Figure 9 here.

In this scenario, we can observe the classical steps of a BCI which are represented as boxes. The measurement of cerebral activity is represented by the "Generic network acquisition" box. The preprocessing step corresponds to two boxes: 1) the "Temporal filter" box, which filters the data in the 3-45 Hz frequency band (here using a butterworth filter) and 2) the "Spatial filter" box, which applies a discrete surface Laplacian filter to the data (Wolpaw et al., 2002) in order to build two Laplacian channels over the left and right motor cortices. The feature extraction step is represented by the "Time based epoching" box, which builds an EEG segment representing the last second of data, each 1/16 second, and by the "Temporal filter", "Simple DSP" and "Signal Average" boxes which are used to compute the BP features in the frequency bands identified in the previous phase. Here the "Simple DSP" box allows us to apply any mathematical formula (such as log-transform or squaring) to the incoming data. These features are then aggregated into a feature vector ("Feature aggregator" box). Note that all these boxes for feature extraction are generated and assembled automatically when running the offline training scenarios, and as such

do not need to be assembled by hand. The “LDA classifier” box is the classification step and uses the LDA trained during the previous phase. Finally, the output of this classifier is sent through the VRPN server (“Analog VRPN Server” box) to the VR application which translates it into a command used to interact with the VE and to provide feedback to the subject. The “XML stimulation player” box is used here to generate the instructions, i.e., which movement (left or right) the subject has to imagine. Instructions are used here in order to measure the subject performances. This box sends events to the VR application, using the “Button VRPN server” box which will then provide the corresponding stimuli to the subject. It should be noted that, as most existing BCI, this BCI is synchronous which means the user can interact with the application only during specific time periods, imposed by the system.

9.1.4 An online experiment with the “Handball” application

In order to illustrate the use of our BCI platform and its suitability to design BCI-based interaction devices for VR, we performed a pilot study with two male subjects (23 and 25 year old). They participated in an online experiment with the “Handball” application in order to assess whether BCI implemented with OpenVIBE could be used to interact with a VR application. The two subjects had previously participated in a few motor imagery BCI experiments. It was however the first time they used this specific BCI-based VR application. The subjects’ brain activity was recorded using 10 EEG channels (FC3, FC4, C5, C3, C1, C2, C4, C6, CP3, CP4),

located over the left and right motor cortices, using a Nexus 32b EEG machine from the Mind Media company. The experiment took place in an immersive virtual reality room equipped with a 3 meters curved wall on which the VE was projected. Subjects were equipped with stereoscopic glasses. The VE was displayed at a frame rate of 96 Hz.

The two subjects first participated in sessions during which the EEG signals were recorded and stored (step 1 above). These EEG signals were then used to train the LDA classifier (step 2 above). Once a suitable classifier was obtained, the two subjects participated in two game sessions each, as described in section 9.1.2 (step 3 above). Subject 1 reached a score of 82.5 % (33/40) in his two game sessions whereas subject 2 reached a performance of 70 % (28/40) for the first session and 87.5 % (35/40) for the second session. By comparison, the score expected by a randomly performing system would be 50 % (20/40). These performances suggest that the subjects were actually having control over the VR application thanks to the BCI. Both subjects also reported that they found the application really entertaining and motivating, which is in line with results from the literature reporting that VR can increase the motivation during BCI experiments (Friedman et al., 2007). Naturally, these results should be moderated by the small number of subjects involved but they still suggest that OpenViBE can be used to design BCI system for interaction with VE. Further evaluations of this application with more subjects is part of ongoing works.

9.2 The “Use-the-force” application.

In addition to the “Handball” VR application, we have developed another VR application based on OpenViBE. This application is known as the “Use-the-force” application, and is an entertaining VR application inspired by the famous “Star wars^{TM}” movie. The aim of this application was to explore the design of self-paced BCI (see below) and to further validate the OpenViBE platform with many users and in real-life conditions, outside laboratories. In the “Use-the-force” application, subjects could lift a virtual space-ship (a “TIE-fighter”) by performing real or imagined foot movements. Indeed, it is well known that, briefly after a real or imagined foot movement, a specific brain signal is generated in the user’s brain: an Event Related Synchronization (ERS) in the Beta rhythm, i.e., a brisk increase of EEG amplitude in the 16-24 Hz frequency band (Pfurtscheller, 1999). Interestingly, this brain signal is mainly located in the central area of the brain, and is therefore potentially detectable with a single electrode (electrode Cz).

Using OpenViBE, we have designed a BCI system that can detect this Beta ERS in electrode Cz, in a self-paced way, i.e., at any time and not only during specific periods. This BCI simply consists of the estimation of a Band Power feature in the 16-24 Hz band, followed by a comparison of this feature value with a threshold in order to detect whether an ERS occured. In the VR application, each time a ERS was detected, the virtual spaceship was lifted up at a speed proportional to the amplitude of the ERS. Figure 10 illustrates this application in action in an

immersive VR room.

We have evaluated this application with 21 subjects who had no previous BCI experience, during a VR exhibition, i.e., in real-life conditions, with a very noisy environment. Our results showed that despite the use of a single electrode and a simple BCI, and despite the fact that the subjects were naive, untrained and in a very noisy environment, more than half of them were able to control the virtual spaceship using real foot movement from the very first time, and similarly a quarter of them could control it using imagined movement, from the very first time. More details about this experiment can be found in (Lotte et al., 2008). In summary, the conducted experiments with this second VR application showed the capability of the OpenViBE platform to design BCI and to use them in real-life conditions. Moreover, regarding the challenging conditions of the experiment (a single EEG channel, no subject training, very noisy environment, etc), the results obtained appeared as very promising.

Figure 10 here.

10 Performance tests

To evaluate OpenViBE performances during online operation, we used two different scenarios that were run on three different hardware configurations. Configuration A is an Intel(R) Xeon(TM) CPU 3.80 GHz computer with 4 GB of RAM and running GNU/Linux Fedora Core 6. Configuration B is an Intel(R) Core(TM)2 CPU T7400 2.16GHz laptop with 2 GB of RAM and running GNU/Linux Fedora Core 5. Configuration C is an Intel(R) Core(TM) 2 DUO CPU E6850 3GHz computer with 4 GB of RAM and running GNU/Linux Ubuntu 9.04 Jaunty.

The first scenario is the Handball VR application scenario which represents a realistic implementation of a BCI. Indeed, the BCI used in the Handball VR application consists of frequency and spatial filtering as preprocessing, followed by feature extraction with band-power estimation, and completed by a LDA as classifier. This design corresponds to well known BCI systems such as the Graz BCI (Ramoser, Muller-Gerking, & Pfurtscheller, 2000; Pfurtscheller & Neuper, 2001) or the Berlin BCI (Blankertz et al., 2006). The main difference between these two BCI and the Handball application BCI lies in the spatial filter used: the Graz-BCI uses bipolar and Laplacian derivations, our BCI uses a Laplacian derivation and the Berlin BCI uses the Common Spatial Patterns (CSP) algorithm. However, these three spatial filters are all simple linear spatial filters and, as such, require similar computation times. Further differences between these three BCI exist in the machine learning algorithms employed to calibrate these BCI. However, as machine learning for BCI calibration is an offline operation, it is not of concern here.

The OpenViBE scenario for the BCI of the handball application is composed of 34 OpenViBE boxes. It consists in processing 11 channels (10 EEG channels + a reference channel) sampled at 512 Hz and acquired in blocks of 32 samples. The signal processing pipeline is identical to the one described in section 9.1.1. The average processor load was computed every second during five minutes (300 measures). The global average over the five minutes is presented in Table 10, for each configuration.

In the second scenario, we tried to reach the limits of the platform. The scenario consisted in reading a 512 channels EEG file followed by multiple butterworth band-pass filters. We added as many band-pass filters as possible while still keeping a processor load below 100 %. Such a scenario could be used when analyzing multiple frequency bands for a large number of channels, e.g., to design a magnetoencephalography-based BCI (Mellinger et al., 2007). Indeed, MEG system are generally composed of hundreds of channels. As in the first scenario, the average processor load was computed every second during five minutes. The number of filters we were able to process in real-time and the associated global processor load average are displayed in Table 10.

Taken together our results suggests that our system is able to address realistic use cases such as a motor imagery based BCI. They also show that our system is able to apply a large number of signal processing algorithms (e.g., band-pass filters) while still keeping the real-time constraints.

Computer configuration	Processor load on scenario 1	Maximum number of filters on scenario 2	Processor load on scenario 2
A	7.17%	6 144	98.26%
B	6.80%	7 680	88.13%
C	3.45%	19 896	97.93%

Table 1: Performance tests: processor load of scenario 1 and maximum number of filters of scenario 2 with corresponding processor load under different hardware configurations.

11 Current state of the platform

The OpenViBE software can be downloaded for free at http://openvibe.inria.fr under the terms of L-GPL[6]. The software currently runs on Microsoft Windows 2000/XP/Vista/7 and GNU/Linux. Several acquisition devices are already supported. Those include for instance Brainamp Standard, g.Tec g.USBamp, MindMedia NeXus32B, MicroMed IntraEEG and CTF/VSF MEG [7].
Existing boxes include generic network acquisition, file reading and writing, signal processing and filtering, feature extraction and basic classifications, in addition to most common visualization paradigms (e.g., raw signals, spectra, time frequency analysis, 2D/3D topography).

[6]Plug-ins relying on GPL are available separately under GPL terms

[7]A complete list of supported devices can be found on the OpenViBE website http://openvibe.inria.fr

It also provides the necessary tools to easily use 3D objects in immersive VE, within the platform, in order to design advanced feedback or visualizations. The software is delivered with ready-to-use scenarios. For instance it proposes a scenario of a BCI based on imagined hand movements that could already be used as an interaction device for various applications in real and virtual environments.

12 Conclusion

In this paper we have presented the OpenViBE platform, a free and open-source platform to design, test and use Brain-Computer Interfaces in real or virtual environments. Our platform provides the necessary tools for real-time data acquisition, processing and display of brain signals. The key features of the platform are 1) a high modularity, 2) embedded tools for visualization and feedback based on VR and 3D displays, 3) BCI design made available to non-programmers thanks to the OpenViBE designer that enables to setup a complete BCI without writing a single line of code and 4) various tools offered to the different types of user, such as the acquisition server or the pre-configured scenarios. OpenViBE also offers the possibility to be used easily as an interaction device with any real or virtual environment. The platform capabilities were illustrated on two entertaining VR applications based on a BCI. In the first application, users could move a virtual ball by imagining hand movements, while in the second

one they could lift a virtual spaceship by using real or imagined foot movements. The evaluation of the platform performances has shown its suitability for real-time applications. We believe that OpenViBE could prove a valuable and useful tool to design innovative BCI-based interaction devices for both VR and real-life applications. This could include applications such as video games and assistive devices. Interested readers can refer to the OpenViBE web site to follow the evolution of the platform and download it for free (INRIA, 2009).

Future work. The OpenViBE platform being an open-source software, it is aimed at being continuously improved and extended, hopefully also by contributors from many different institutions. Currently, our labs are working to propose new functionalities for the authoring tools in order to increase the productivity of end users (both for authors and operators). Efforts will be also dedicated to the growth of the platform by adding, for example new algorithms for detecting more efficiently a higher number of signals or new visualization techniques of brain activity in VR. Lastly, distributed computing over dedicated hardware will be implemented in the kernel thanks to the box concept.

Acknowledgments

This work was supported by the French National Research Agency within the OpenViBE project and grant ANR05RNTL01601. During this work, Fabien Lotte was at INRIA Rennes, France. He

now is at the Institute for Infocomm Research (I2R), Singapore. The authors would like to thank Dr. Marc Christie for his fruitful comments on this manuscript.

References

Arrouët, C., Congedo, M., Marvie, J. E., Lamarche, F., Lecuyer, A., & Arnaldi, B. (2005). Open-ViBE : a 3D platform for real-time neuroscience. *Journal of Neurotherapy*, *9*(1), 3-25.

Baillet, S., Mosher, J., & Leahy, R. (2001). Electromagnetic brain mapping. *IEEE Signal Processing Magazine*, *18*(6), 14-30.

Bashashati, A., Fatourechi, M., Ward, R. K., & Birch, G. E. (2007). A survey of signal processing algorithms in brain-computer interfaces based on electrical brain signals. *Journal of Neural engineering*, *4*(2), R35-57.

Blankertz, B., Dornhege, G., Krauledat, M., Mller, K.-R., Kunzmann, V., Losch, F., et al. (2006). The Berlin brain-computer interface: EEG-based communication without subject training. *IEEE Trans. Neural Sys. Rehab. Eng.*, *14*(2), 147-152.

Burdea, G. (1996). *Force and touch feedback for virtual reality*. John Wiley & Sons, Inc. New York, NY, USA.

Donchin, E., Spencer, K., & Wijesinghe, R. (2000). The mental prosthesis: assessing the speed of

a P300-based brain-computer interface. *IEEE Transactions on Rehabilitation Engineering*, *8*(2), 174-179.

Farwell, L., & Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, *70*, 510-523.

Friedman, D., Leeb, R., Guger, C., Steed, A., Pfurtscheller, G., & Slater, M. (2007). Navigating virtual reality by thought: What is it like? *Presence*, *16*(1), 100-110.

INRIA. (2009). *Open-ViBE platform website.* (http://openvibe.inria.fr/)

Krepki, R., Blankertz, B., Curio, G., & Müller, K. R. (2007). The berlin brain-computer interface (BBCI): towards a new communication channel for online control in gaming applications. *Journal of Multimedia Tools and Applications*, *33*(1), 73-90.

Lécuyer, A., Lotte, F., Reilly, R., Leeb, R., Hirose, M., & Slater, M. (2008). Brain-computer interfaces, virtual reality and videogames. *IEEE Computer*, *41*(10), 66-72.

Leeb, R., Keinrath, C., Friedman, D., Guger, C., Scherer, R., Neuper, C., et al. (2006). Walking by thinking: The brainwaves are crucial, not the muscles! *Presence: Teleoperators and Virtual Environments*, *15*(5), 500-51.

Leeb, R., Scherer, R., Friedman, D., Lee, F., Keinrath, C., Bischof, H., et al. (2007). Towards brain-computer interfacing. In (G. Dornhege, R. Millan Jdel, T. Hinterberger, D. J. McFarland & K. R. Müller ed., chap. Combining BCI and Virtual Reality: Scouting Virtual

Worlds). MIT Press.

Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., & Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of Neural Engineering*, *4*, R1-R13.

Lotte, F., Lécuyer, A., Lamarche, F., & Arnaldi, B. (2007). Studying the use of fuzzy inference systems for motor imagery classification. *IEEE Transactions on Neural System and Rehabilitation Engineering*, *15*(2), 322-324.

Lotte, F., Renard, Y., & Lécuyer, A. (2008). Self-paced brain-computer interaction with virtual worlds: a qualitative and quantitative study 'out-of-the-lab'. In *4th international brain-computer interface workshop and training course* (p. 373-378).

Maggi, L., Parini, S., Perego, P., & Andreoni, G. (2008). BCI++: an object-oriented BCI prototyping framework. In *4th international brain-computer interface workshop 2008.*

Mellinger, J., & Schalk, G. (2007). Toward brain-computer interfacing. In (In: G. Dornhege, J.R. Milln et al. (eds.) ed., p. 359-368). MIT Press.

Mellinger, J., Schalk, G., Braun, C., Preissl, H., Rosenstiel, W., Birbaumer, N., et al. (2007). An MEG-based brain-computer interface (BCI). *Neuroimage*, *36*(3), 581-593.

Millán, J. (2008). Brain-controlled robots. *IEEE Intelligent Systems*.

Neuper, C., Scherer, R., Wriessnegger, S., & Pfurtscheller, G. (2009). Motor imagery and action observation: Modulation of sensorimotor brain rhythms during mental control of a

braincomputer interface. *Clinical Neurophysiology*, *120*(2), 239-247.

Nijholt, A. (2009). BCI for games: A 'state of the art' survey. In *Icec'2009* (p. 225-228).

Pfurtscheller, G. (1999). EEG event-related desynchronization (ERD) and event-related synchronization (ERS). *Electroencephalography: Basic Principles, Clinical Applications and Related Fields, 4th ed.*, 958,967.

Pfurtscheller, G., & Neuper, C. (2001). Motor imagery and direct brain-computer communication. *proceedings of the IEEE*, *89*(7), 1123-1134.

Ramoser, H., Muller-Gerking, J., & Pfurtscheller, G. (2000). Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering*, *8*(4), 441-446.

Sauvan, J., Lécuyer, A., Lotte, F., & Casiez, G. (2009). A performance model of selection techniques for p300-based brain-computer interfaces. In *Acm sigchi conference on human factors in computing systems (acm chi)* (p. 2205-2208).

Schlogl, A. (2006). *GDF - a general dataformat for biosignals.*

Schlögl, A., Brunner, C., Scherer, R., & Glatz, A. (2007). Towards brain-computer interfacing. In (G. Dornhege, J.R. Millan, T. Hinterberger, D.J. McFarland, K.-R. Mller ed., p. 347-358). MIT press.

Taylor, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., & Helser, A. (2001). VRPN: a device-independent, network-transparent VR peripheral system. In *Vrst '01: Proceedings*

of the acm symposium on virtual reality software and technology (pp. 55–61). New York, NY, USA: ACM Press.

Virtools website. (2007). (http://www.virtools.com/)

Wolpaw, J., Birbaumer, N., McFarland, D., Pfurtscheller, G., & Vaughan, T. (2002). Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, *113*(6), 767-791.

Zhong, M., Lotte, F., Girolami, M., & Lécuyer, A. (2008). Classifying EEG for brain computer interfaces using gaussian processes. *Pattern Recognition Letters*, *29*, 354-359.

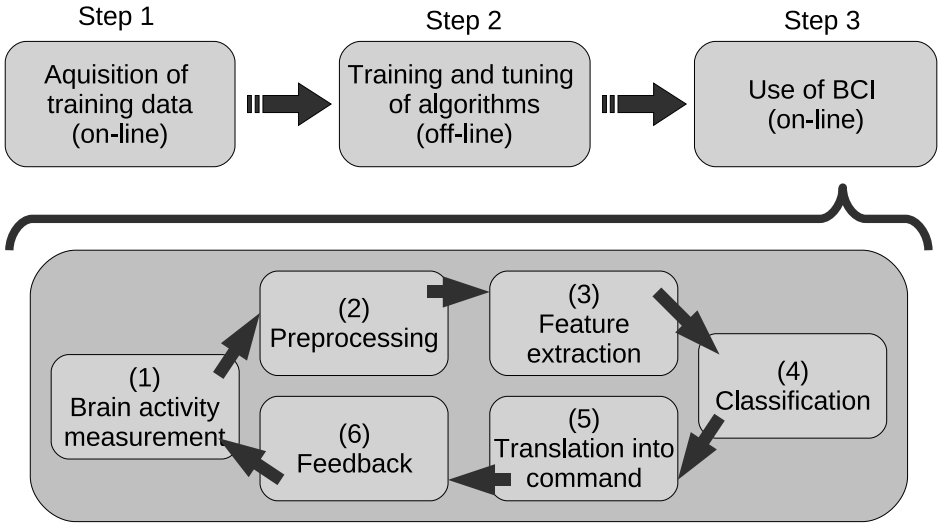
Step 1
Aquisition of training data (on-line)
Step 2
Training and tuning of algorithms (off-line)
Step 3
Use of BCI (on-line)
(1) Brain activity measurement
(2) Preprocessing
(3) Feature extraction
(4) Classification
(5) Translation into command
(6) Feedback

Figure 1:

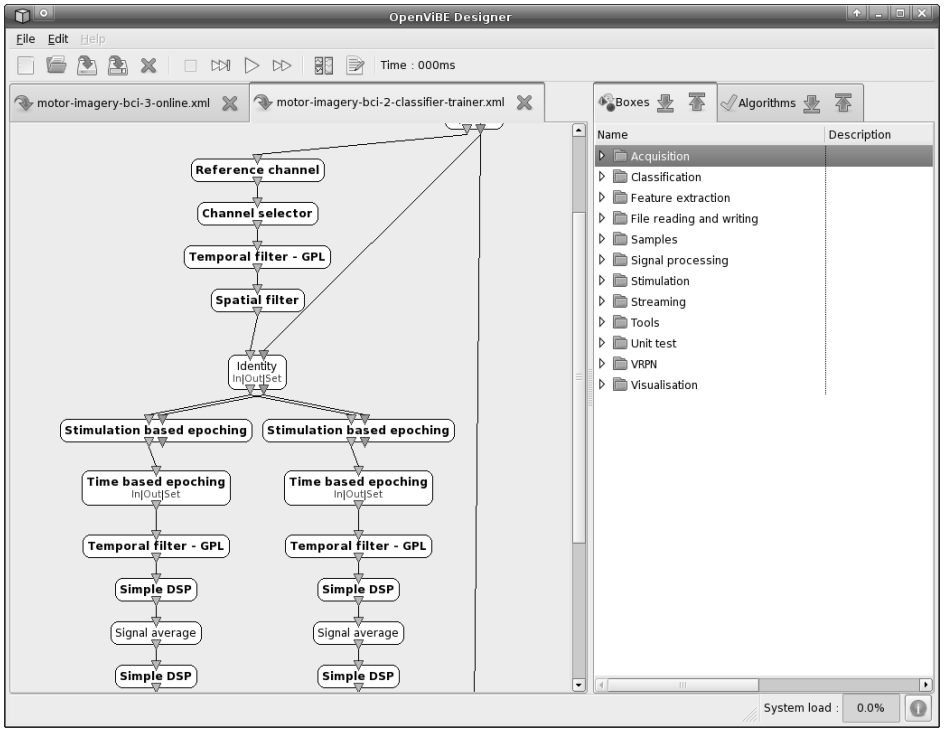
OpenViBE Designer
File Edit Help
Time : 000ms
motor-imagery-bci-3-online.xml
motor-imagery-bci-2-classifier-trainer.xml
Reference channel
Channel selector
Temporal filter - GPL
Spatial filter
Identity
In|Out|Set
Stimulation based epoching
Stimulation based epoching
Time based epoching
In|Out|Set
Time based epoching
In|Out|Set
Temporal filter - GPL
Temporal filter - GPL
Simple DSP
Simple DSP
Signal average
Signal average
Simple DSP
Simple DSP
Boxes
Algorithms
Name
Description
Acquisition
Classification
Feature extraction
File reading and writing
Samples
Signal processing
Stimulation
Streaming
Tools
Unit test
VRPN
Visualisation
System load :
0.0%

Figure 2:

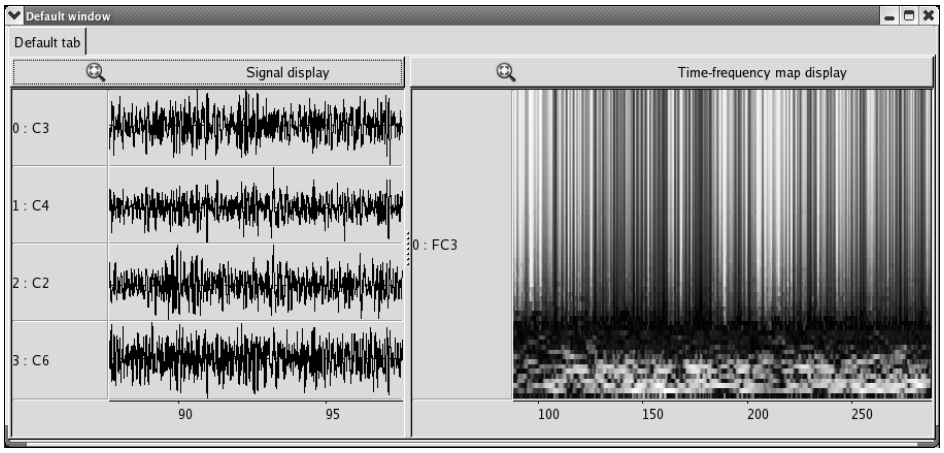
Default window
Default tab
Signal display
Time-frequency map display
0 : C3
1 : C4
2 : C2
3 : C6
0 : FC3
90
95
100
150
200
250

Figure 3:

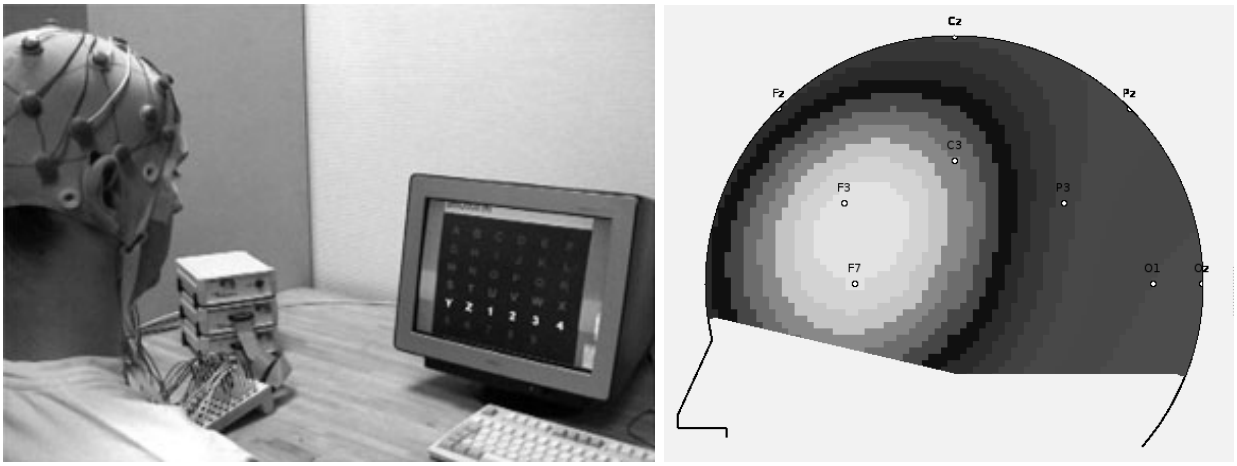

Figure 4:

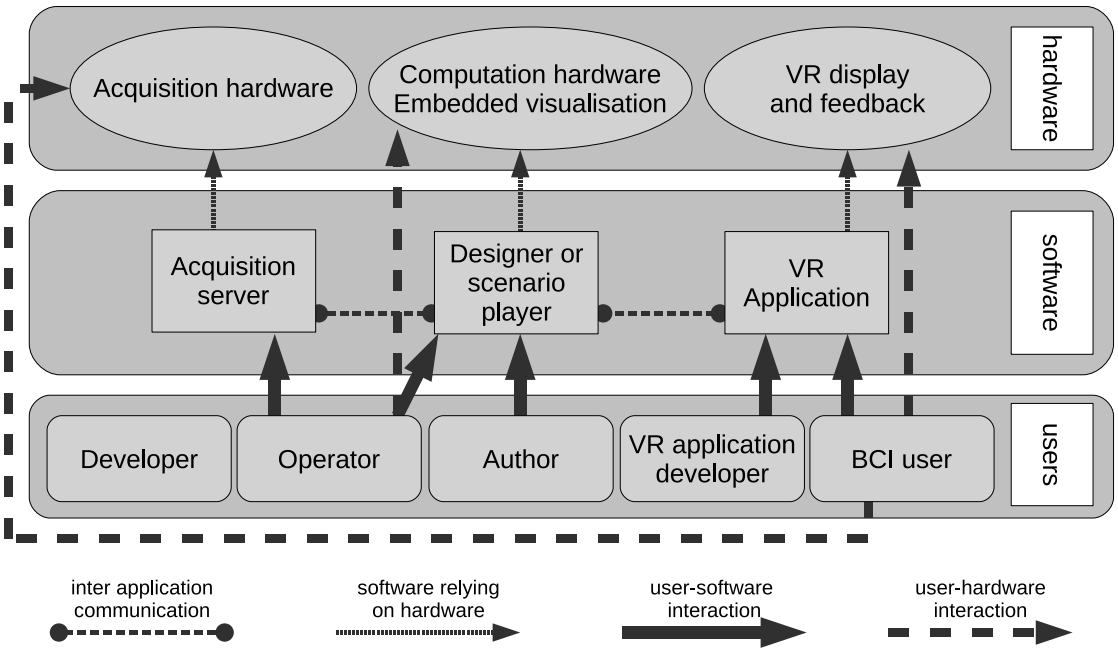
Acquisition hardware
Computation hardware
Embedded visualisation
VR display
and feedback
hardware
Acquisition
server
Designer or
scenario
player
VR
Application
software
Developer
Operator
Author
VR application
developer
BCI user
users
inter application
communication
software relying
on hardware
user-software
interaction
user-hardware
interaction

Figure 5:

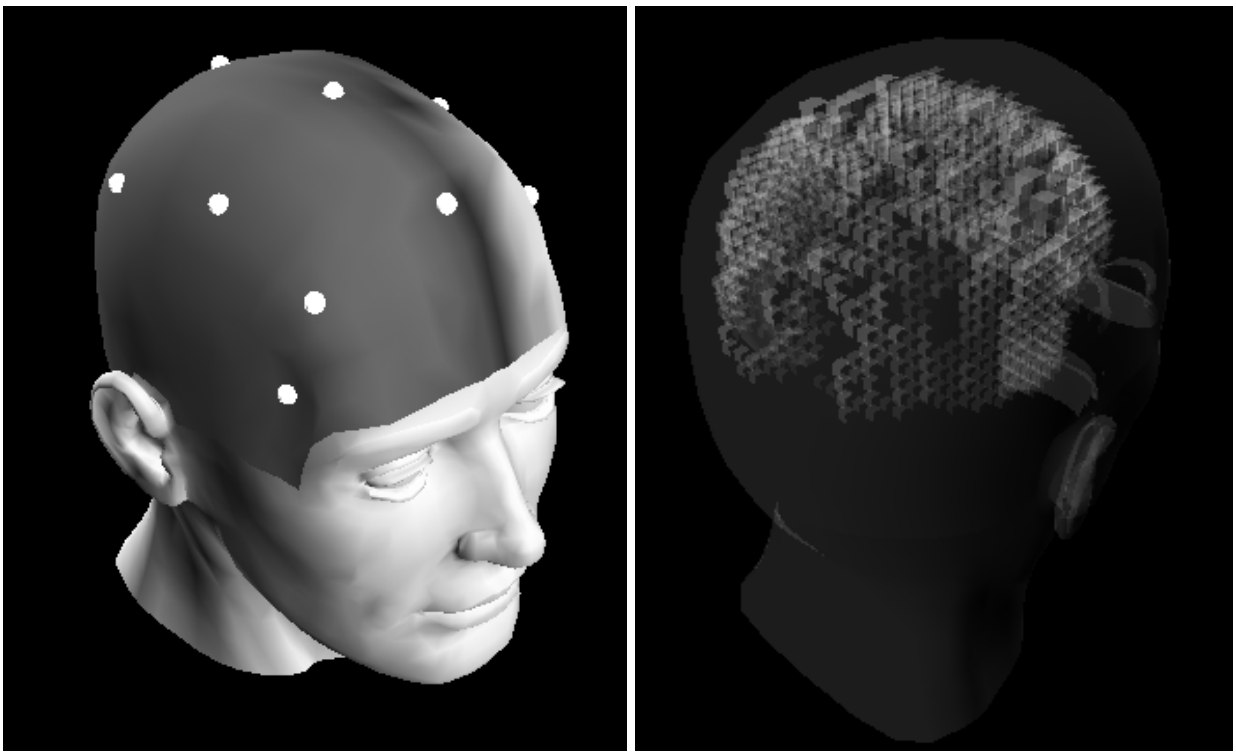

Figure 6:

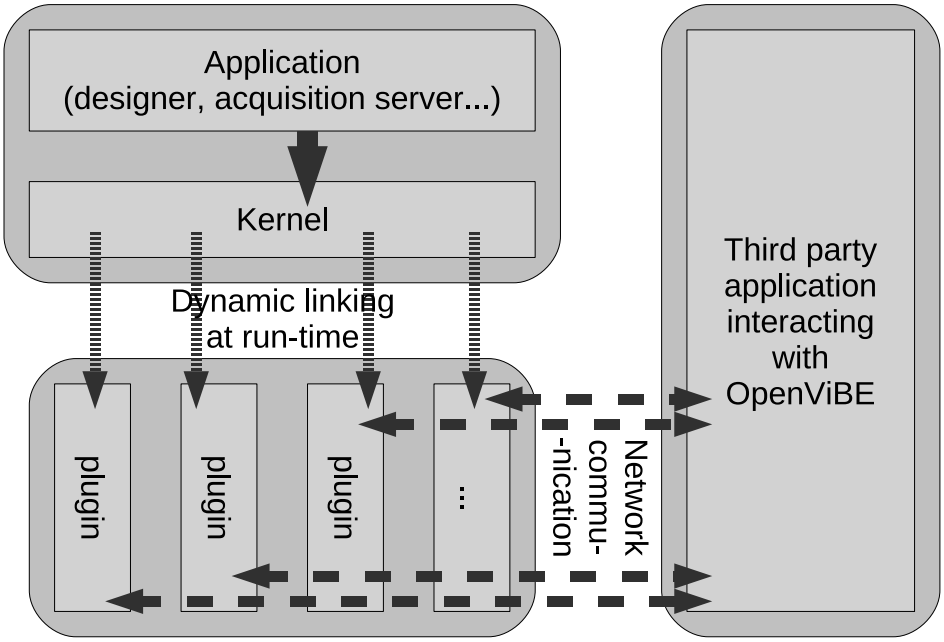
Application
(designer, acquisition server...)
Kernel
Dynamic linking
at run-time
plugin
plugin
plugin
...
Network
commu-
-nication
Third party
application
interacting
with
OpenViBE

Figure 7:

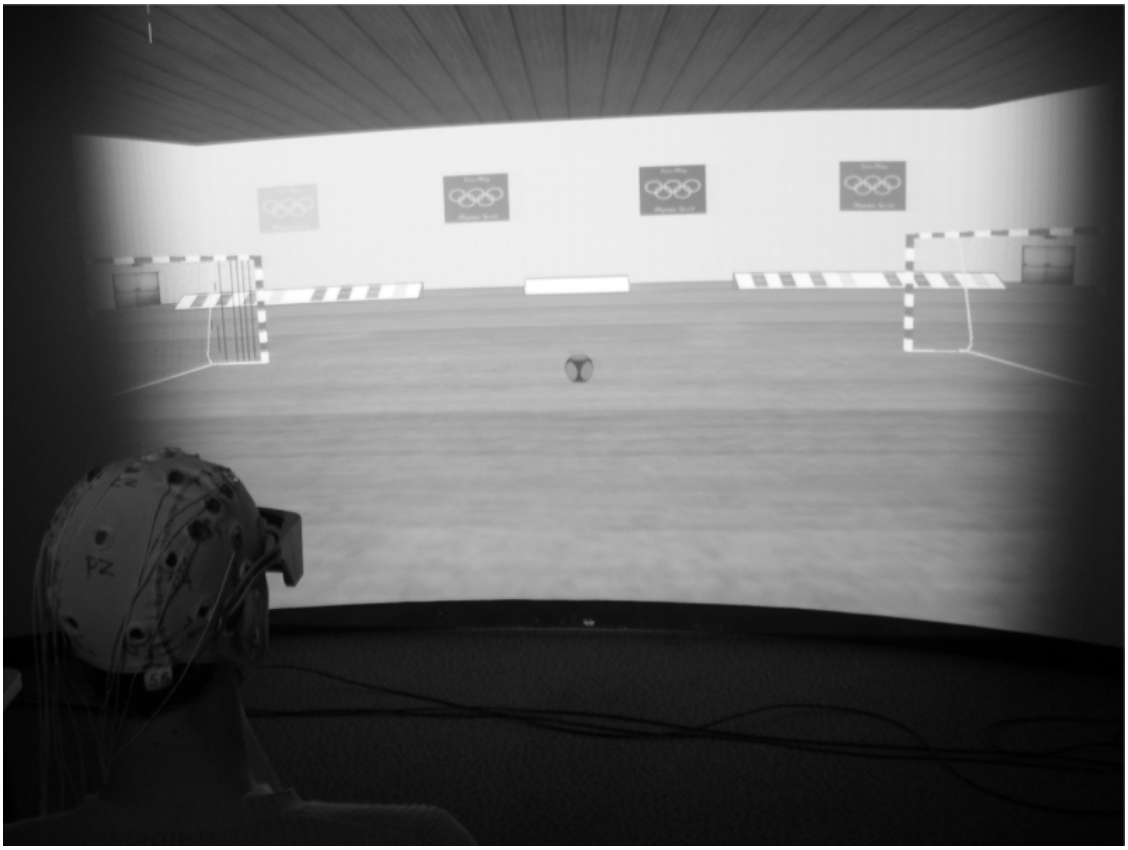

Figure 8:

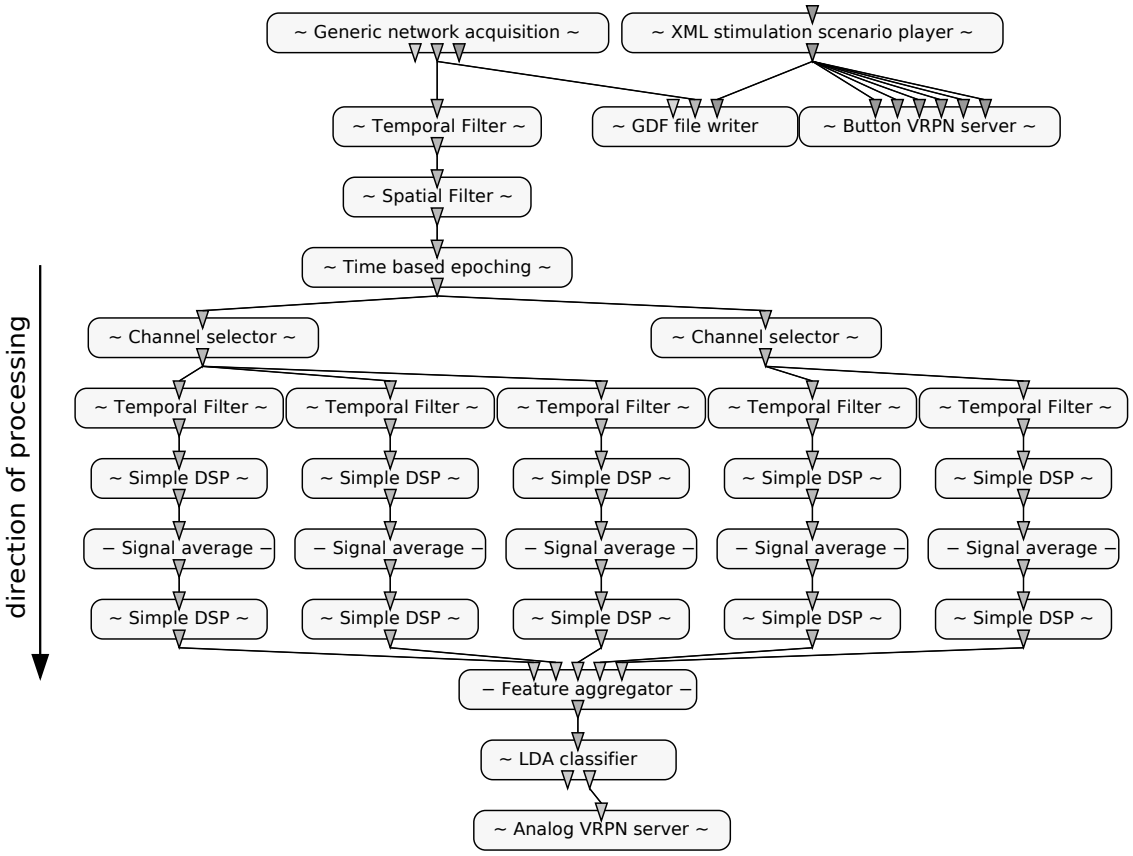
~ Generic network acquisition ~
~ XML stimulation scenario player ~
~ Temporal Filter ~
~ GDF file writer
~ Button VRPN server ~
~ Spatial Filter ~
~ Time based epoching ~
direction of processing
~ Channel selector ~
~ Channel selector ~
~ Temporal Filter ~
~ Temporal Filter ~
~ Temporal Filter ~
~ Temporal Filter ~
~ Temporal Filter ~
~ Simple DSP ~
~ Simple DSP ~
~ Simple DSP ~
~ Simple DSP ~
~ Simple DSP ~
– Signal average –
– Signal average –
– Signal average –
– Signal average –
– Signal average –
~ Simple DSP ~
~ Simple DSP ~
~ Simple DSP ~
~ Simple DSP ~
~ Simple DSP ~
– Feature aggregator –
~ LDA classifier
~ Analog VRPN server ~

Figure 9:

Figure 10:

Figure captions

Figure 1: Designing a BCI with OpenViBE. **One-column width.**

Figure 2: The OpenViBE designer with a sample scenario. The tool enables the graphical design of a BCI system by adding and connecting boxes representing processing modules without writing a single line of code. **Two-column width.**

Figure 3: Examples of 2D displays: raw signals and time-frequency map display widgets. **One-column width.**

Figure 4: Examples of visualization widgets available in OpenViBE. Left: The P300-speller. Right: 2D visualization of brain activity in real-time, on the scalp. **One-column width.**

Figure 5: Relations between users, hardware and software components. **Two-columns width.**

Figure 6: 3D display of brain activity. Left: 3D topographic display of brain activity in real-time, on the scalp. Right: Voxel reconstruction of brain activity inside the brain, based on scalp measures. **One-column width.**

Figure 7: Software architecture. **One-column width.**

Figure 8: The Handball VR application. The user can move the virtual ball towards the left or right by imagining left or right hand movements. **One-column width.**

Figure 9: OpenViBE scenario for the Handball application. This scenario performs the online processing of the recorded EEG data in order to identify left or right imagined hand movements. The output of this processing is sent to the VR application by using VRPN. **Two-column width.**

Figure 10: The "Use-the-force" application. In this application, the user can lift a virtual spaceship by performing real or imagined foot movements. (©CNRS Photothèque/Hubert Raguet). **One-column width.**