BCI Software Platforms

C. Brunner, G. Andreoni, L. Bianchi, B. Blankertz, C. Breitwieser, S. Kanoh, C. A. Kothe, A. Lécuyer, S. Makeig, J. Mellinger, P. Perego, Y. Renard, G. Schalk, I. P. Susila, B. Venthur, and G. R. Müller-Putz

Abstract In this chapter, we provide an overview of publicly available software platforms for brain-computer interfaces. We have identified seven major BCI platforms and one platform specifically targeted towards feedback and stimulus presentation. We describe the intended target user group (which includes researchers, programmers, and end users), the most important features of each platform such as availability on different operating systems, licences, programming languages involved, supported devices, and so on. These seven platforms are: (1) BCI2000, (2) OpenViBE, (3) TOBI Common Implementation Platform (CIP), (4) BCILAB, (5)

C. Brunner, C. Breitwieser, G. R. Müller-Putz
Institute for Knowledge Discovery, Graz University of Technology, Austria, e-mail: clemens.brunner@tugraz.at

C. Brunner, C. A. Kothe, S. Makeig
Swartz Center for Computational Neuroscience, INC, UCSD, San Diego, CA, USA

G. Andreoni, P. Perego
INDACO, Politecnico di Milano, Milan, Italy

L. Bianchi
Neuroscience Department, Tor Vergata University of Rome, Rome, Italy

B. Blankertz, B. Venthur
Machine Learning Laboratory, Berlin Institute of Technology, Berlin, Germany

S. Kanoh
Department of Electronics and Intelligent Systems, Tohoku Institute of Technology, Sendai, Japan

J. Mellinger
Institute of Medical Psychology and Behavioral Neurobiology, University of Tübingen, Germany

Y. Renard, A. Lécuyer
National Institute for Research in Computer Science and Control (INRIA), Rennes, France

G. Schalk
Wadsworth Center, New York State Department of Health, Albany, NY, USA

I. P. Susila
Nuclear Equipment Engineering Center, Tangerang Selatan, Indonesia

BCI++, (6) xBCI, and (7) BF++. The feedback framework is called Pyff. Our conclusion discusses possible synergies and future developments, such as combining different components of different platforms. With this overview, we hope to identify the strengths and weaknesses of each platform on the market, which should help anyone in the BCI research field in their decision which platform to use for their specific purposes.

1 Introduction

Research on brain-computer interfaces (BCIs) started as early as 1973, when Jacques J. Vidal presented the first concept of direct brain-computer communication [54]. Since then, many research groups have developed this first idea into functional prototypes. While there are still many open issues that need to be addressed, the first BCIs are already being used outside the labs, for example in hospitals or at homes [52, 35, 48].

With the advent of modern personal computers, computational power was more than sufficient for most BCI requirements. Moreover, more user-friendly development environments started to emerge, and applications started to rely heavily on graphical representations of objects and relationships between objects. For example, the combination of MATLAB and Simulink (The Mathworks, Inc.) is probably one of the most popular commercial general-purpose platforms for developing a great variety of different scientific applications.

Software platforms specifically targeted towards the development of BCIs should offer frequently used building blocks such as data acquisition, feature extraction, classification, and feedback presentation modules. Many labs have developed their own custom set of tools over many years, based on different requirements, programming languages, and prospective users. These tools are typically closed source and not available to the public, since they are primarily used for rapid prototyping and in-house testing. Moreover, such tools might lack extensive documentation and might not be readily useable for others outside the lab.

On the other hand, several publicly available BCI platforms have been released during the past few years. These frameworks are targeted either towards BCI developers, BCI users, or both. Some platforms are released under popular open source licenses (such as the GNU General Public License[1]), which allow everyone to inspect, modify, and redistribute the source code. Moreover, many frameworks are cross-platform, which means that they can be deployed on several different operating systems, whereas others are restricted to either a specific operating system and/or require commercial software.

This article provides an overview of currently available platforms and frameworks for developing and deploying BCIs. We have identified seven major BCI platforms and one platform specifically targeted towards feedback and stimulus pre-

[1] www.gnu.org/licenses/gpl.html

sentation. These platforms are: (1) BCI2000, (2) OpenViBE, (3) TOBI Common Implementation Platform (CIP), (4) BCILAB, (5) BCI++, (6) xBCI, and (7) BF++. The framework for feedback and stimulus presentation is called Pyff and does not have any direct competitors at the moment. Among the seven platforms, TOBI CIP plays a special role, because it is not a full-fledged BCI platform. Instead, this platform defines standardized interfaces between different BCI components. This allows other BCI platforms that implement specific modules (such as data acquisition, feature extraction or classification blocks) which adhere to the TOBI CIP specifications to talk to and interact with each other.

2 BCI2000

BCI2000[2] is a general-purpose software platform for BCI research. It can also be used for a wide variety of data acquisition, stimulus presentation, and brain monitoring applications. BCI2000 has been in development since 2000 in a project led by the Brain-Computer Interface R&D Program at the Wadsworth Center of the New York State Department of Health in Albany, New York, USA, with substantial contributions by the Institute of Medical Psychology and Behavioral Neurobiology at the University of Tübingen, Germany. In addition, many laboratories around the world, most notably the BrainLab at Georgia Tech in Atlanta, Georgia, and Fondazione Santa Lucia in Rome, Italy, have also played an important role in the project's development. BCI2000 is currently maintained and further developed by a core team consisting of six scientists and programmers, and by a community of contributors that constantly expand the capabilities of the system, such as by adding support for new hardware devices. The BCI2000 core team consists of Gerwin Schalk (Project Director and Chief Evangelist), Jürgen Mellinger (Chief Software Engineer), Jeremy Hill (Project Coordinator), Griffin Milsap (Software and Test Engineer), Adam Wilson (User Management and Support), and Peter Brunner (Workshops and Tutorials).

BCI2000 has several characteristics that are important to mention. These characteristics are described in the following paragraphs.

An Established Solution BCI2000 comes with support for different data acquisition hardware, signal processing routines, and experimental paradigms. Specifically, BCI2000 supports 19 different data acquisition systems by different manufacturers, including all major digital EEG amplifiers. It supports appropriate processing of EEG oscillations, evoked potentials, ECoG activity, and single-unit action potentials. The resulting outputs can control cursor movement and provide spelling. BCI2000 can also provide highly customizable auditory/visual stimulation that is synchronized with acquisition of brain signals and other inputs (such as eye trackers).

[2] `www.bci2000.org`

Comprehensive Documentation BCI2000 provides documentation for both researchers and programmers. Documentation for researchers describes how to operate and configure existing BCI2000 components. Documentation for programmers describes the data structures, data types, and internal events in the BCI2000 online system. It also describes how to extend BCI2000 with new acquisition modules, signal processing components, or application modules. For both researchers and programmers, information is available in the form of tutorials as well as detailed references. BCI2000 documentation is provided with each BCI2000 installation and is also available online[3]. In addition, there is a bulletin board[4] for questions about BCI2000 and BCI systems in general. Finally, there is a book on the BCI2000 system, which includes an introduction to all major aspects of BCI operation [46].

Programming Languages and Compatibility BCI2000 has been written in C++, and is thus very efficient in terms of resource utilization. It provides a programming interface that allows to access system parameters, data signals, and event information in a concise and intuitive way. In addition, BCI2000 provides support for writing online signal processing code in MATLAB, and includes an entire layer of Python compatibility. This Python layer allows for writing complete BCI2000 modules that support data acquisition, signal processing, or application output. For compatibility with even more programming languages and external applications, BCI2000's core functionality comes as a loadable library, and may be wrapped into an application that accesses this library. Furthermore, BCI2000 exposes its internal state over a socket interface based on UDP (user datagram protocol), which can be read and written to by an external application. For code compilation, BCI2000 supports Visual Studio – including the freely available Express versions – and GCC[5]/MinGW[6] in addition to the Borland C++ compiler it was originally developed with. This set of compilers allows compilation of BCI2000 on multiple platforms, including Windows and Mac OS X. At the same time, BCI2000 is currently fully tested and supported only on Windows. Finally, BCI2000 is freely available under the terms of the GNU General Public License.

Simple Deployment The BCI2000 platform does not rely on third-party software components. A full BCI2000 installation is contained in a single directory tree. BCI2000 may be deployed simply by copying this tree, without the need of administrative rights, and without the need to install additional software. Maintenance of BCI2000 installations across multiple research sites is as easy as synchronizing a centrally maintained installation between sites.

Real-time Performance BCI2000 is usually executed on Microsoft Windows operating systems. Windows does not have dedicated support for real-time operation. However, BCI2000 has very favorable timing behavior that is well suited for BCI experiments. Generally, stimulus and feedback presentation is delivered with mil-

[3] `doc.bci2000.org`

[4] `bbs.bci2000.org`

[5] `gcc.gnu.org`

[6] `www.mingw.org`

lisecond accuracy [56]. BCI2000 comes with a tool that comprehensively characterizes timing behavior for different BCI2000 configurations.

Impact BCI2000 has had a substantial impact on BCI and related research. As of April 2011, BCI2000 has been acquired by more than 900 users around the world. The original article that described the BCI2000 system [45] has been cited close to 400 times (Google Scholar, 4/29/11), and was awarded a Best Paper Award by IEEE Transactions on Biomedical Engineering. Furthermore, a review of the literature revealed that BCI2000 has been used in studies reported in more than 150 peer-reviewed publications. These publications include some of the most impressive BCI demonstrations and applications reported to date such as: the first online brain-computer interfaces using magnetoencephalographic (MEG) signals [31] or electrocorticographic (ECoG) signals [24, 23, 55, 17]; the first multi-dimensional BCI using ECoG signals [47]; the fastest BCI ever demonstrated in humans [9]; the first applications of BCI technology toward restoration of function in patients with chronic stroke [11, 57]; the use of BCI techniques to control assistive technologies [14]; the first real-time BCI use of high-resolution EEG techniques [13]; the first tactile P300 BCI [8]; demonstrations that non-invasive BCI systems can support multi-dimensional cursor movements without [58, 59, 30] and with [29] selection capabilities; control of a humanoid robot by a noninvasive BCI [3]; and the first demonstration that people severely paralyzed by amyotrophic lateral sclerosis (ALS) can operate a sensorimotor rhythm-based BCI [20]. BCI2000 is also supporting the only existing long-term in-home application of BCI technology for people who are severely disabled [48].

Many studies have used BCI2000 in fields related to BCI research. This includes the first large-scale motor mapping studies using ECoG signals [22, 33]; real-time mapping of cortical function using ECoG [32, 44, 10]; the optimization of BCI signal processing routines [60, 12, 42]; evaluation of steady-state visual evoked potentials (SSVEP) for BCI purposes [1]; the demonstration that two-dimensional hand movements and finger movements can be decoded from ECoG signals [43, 19]; and determination of the electrical properties of the dura and its influence on ECoG recordings [51]. Facilitated by the easy exchange of data and experimental paradigms that BCI2000 enables, a number of these studies were performed as collaborations among several geographically widespread laboratories.

Furthermore, BCI2000 has also been used in demonstrations on national US television including NBC, CBS, and CNN; referenced several hundred times in journal articles, media articles, and personal blogs; used or cited in dozens of Masters Theses or Doctoral Dissertations; listed as a significant qualification in curricula vitae; and even mentioned as desirable experience in job postings. The widespread and continually growing success of the BCI2000 platform is strong evidence for its utility.

3 OpenViBE

OpenViBE[7] is a free and open-source software platform for designing, testing, and using brain-computer interfaces. The platform consists of a set of software modules that can be easily and efficiently integrated to develop fully functional BCIs. Key features of the platform are described in the following paragraphs.

Modularity and Reusability OpenViBE consists of a set of software modules devoted to the acquisition, preprocessing, processing, and visualization of cerebral data. The platform also has modules which handle the interaction with applications. OpenViBE is a general purpose platform and allows users to easily add new software modules specifically tailored towards their needs. This is largely made possible by the box concept, an elementary component in the processing pipeline that allows reusable components, reduces development time, and helps to quickly extend functionality.

Different User Types OpenViBE is designed for different types of users, including researchers, developers, and clinicians. Their various needs are addressed and different tools are proposed for each user type, depending on their programming skills and their knowledge of brain physiology. More specifically, OpenViBE includes four types of users. "Developers" and "application developers" are both programmers. "Authors" and "operators" do not need any programming skills. These different types of users differ in the way they can modify and work with the tools provided by OpenViBE.

- A "developer" can extend OpenViBE. To that end, OpenViBE comes with a complete software development kit (SDK). This SDK provides access to functionalities at different levels.
- An "application developer" uses the SDK to create standalone applications which communicate with OpenViBE. Such applications range from the designer authoring tool to external VR applications which a BCI user interacts with.
- An "author" uses the designer application to arrange existing boxes to create a scenario. An author configures these boxes and the scenario to produce a complete and ready-to-use BCI system. An author is aware of the internals of the platform, has a basic knowledge of BCI systems in general, and is familiar with basic signal processing theory. An author is also aware of which interaction paradigm to use. However, authors do not need strong computer programming skills, because they use dedicated tools to perform all tasks.
- An "operator" is a clinician or a practitioner, for example. Consequently, an operator is neither a computer expert nor an OpenViBE expert. Operators are in charge of using and running pre-built scenarios. They know how the BCI system works and monitor the execution of the BCI system with dedicated visualization components. Operators understand neurophysiological signals and can support BCI users to improve their control over the BCI system.

[7] `openvibe.inria.fr`

- Finally, the BCI "user" generally wears the brain activity acquisition hardware (for example, an EEG cap) and interacts with an application by mental activity. The application could be a neurofeedback training program, a video game in virtual reality, a remote operation in augmented reality, and so on. While users do not directly use the OpenViBE platform, they implicitly take advantage of its features.

Portability The OpenViBE platform operates independently from different software targets and hardware devices. It includes an abstract layer of representation, which supports various acquisition devices such as EEG or MEG amplifiers. OpenViBE runs on Windows and Linux platforms. OpenViBE is based on free and portable software such as GTK+[8], IT++[9], VRPN[10], and GCC.

Connection with External Applications OpenViBE can be easily integrated with high-level applications such as virtual reality environments. OpenViBE acts as an external peripheral device to any kind of real or virtual environment. It also takes advantage of virtual reality displays through a scenegraph management library, allowing the visualization of cerebral activity in an intuitive way or the creation of incentive training environments for neurofeedback applications.

OpenViBE Tools The OpenViBE platform includes a large number of useful tools: the acquisition server, the designer, 2D visualization tools, and sample scenarios for BCIs or neurofeedback applications.

The acquisition server provides a generic interface to various kinds of acquisition devices. It allows an author to create hardware-independent scenarios with a generic acquisition box. This box receives data over the network from the acquisition server, which is connected to the hardware and transforms the recorded data in a generic way. The way the acquisition server is connected to the device mostly depends on the hardware manufacturer's tools to access the device. Some devices are shipped with a dedicated SDK, whereas others involve a communication protocol over the network, serial interface or a USB connection.

The designer makes it possible to create complete scenarios using a dedicated graphical language (see Figure 1 left). The user can drag and drop existing modules from a panel to the scenario window. Each module appears as a rectangular box with inputs, outputs, and a dedicated configuration panel. Boxes can be connected through their inputs and outputs. The designer also allows to configure the arrangement of visualization windows. Finally, an embedded player engine supports testing and debugging the current scenario in real time.

The visualization features of OpenViBE are available as specific boxes and include 2D/3D brain activity plots. These boxes can access all functions, and in particular the whole stream content of the connected inputs. OpenViBE offers a wide range of visualization widgets such as raw signal display, gauges, power spectrum, time-frequency maps, and 2D/3D topography (where EEG activity is projected on

[8] `www.gtk.org`

[9] `sourceforge.net/apps/wordpress/itpp`

[10] `www.cs.unc.edu/Research/vrpn`

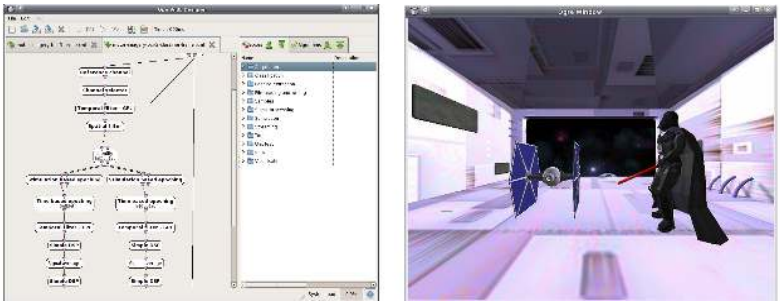

Fig. 1 Left: The OpenViBE designer supports intuitive graphical development of a BCI system. Right: Video game based on motor imagery using a self-paced BCI developed with OpenViBE [25].

the scalp surface). OpenViBE also provides presentation widgets that display instructions to a user, for example as used in typical BCI paradigms such as the classical cue-based motor imagery paradigm or the P300 speller.

Existing and pre-configured ready-to-use scenarios are provided to assist the user. Currently, six scenarios are available:

- The motor imagery based BCI scenario uses OpenViBE as an interaction peripheral device with imagined movements of the left and right hands.
- The Self-paced BCI scenario implements a BCI based on real or imagined foot movements in a self-paced way (Figure 1 right).
- The neurofeedback scenario displays the power of the brain activity in a specific frequency band for neurofeedback applications.
- The real time visualization scenario visualizes brain activity of a user in real time on a 2D or 3D head model. This scenario can be used together with inverse solution methods to visualize brain activity in the whole brain volume in addition to the scalp surface.
- The P300 speller scenario implements the famous P300 speller, a BCI used to spell letters by using the P300 component of visual event-related potentials.
- The SSVEP scenario allows a user to control a simple game by focusing on flickering targets on the screen. The scenario detects SSVEP at occipital sites to move a virtual object.

Development Team and Community OpenViBE is licensed under the GNU Lesser General Public License (version 2 or later)[11]. It is officially available for Microsoft Windows (XP to 7) and Linux (Ubuntu and Fedora) platforms. Other operating systems have been addressed by the community. OpenViBE is released every three months by the French National Institute for Research in Computer Science and Control (INRIA). The core development team at INRIA consists of three engineers who focus on new features, integration of community contributions, and releases. People who contributed to OpenViBE include A. Lécuyer, Y. Renard,

[11] `www.gnu.org/copyleft/lesser.html`

F. Lotte, L. Bougrain, L. Bonnet, J. Legény, V. Delannoy, B. Payan, M. Clerc, T. Papadopoulo, and J. Fruitet (INRIA); O. Bertrand, J.-P. Lachaux, G. Gibert, E. Maby, and J. Mattout (INSERM); M. Congedo, G. Ionescu, M. Goyat, G. Lio, and N. Tarrin (GIPSA-LAB); and A. Souloumiac and B. Rivet (CEA).

It is difficult to reliably estimate the number of OpenViBE users, because OpenViBE can be downloaded and used without any kind of registration. However, the OpenViBE Windows installer has been downloaded more than 300 times a month in 2010, and the OpenViBE website has been visited by more than 3000 single visitors per month. The non-exhaustive list of identified users of OpenViBE is provided on the OpenViBE website and includes many universities, research institutes, and medical centers all around the world. OpenViBE is also used in a large variety of projects involving industrial or medical partners, for example in video games or assistance to disabled people.

4 TOBI

The TOBI Common Implementation Platform (CIP)[12] is a cross-platform set of interfaces which connect parts of different BCI systems. These interfaces transmit raw data, extracted features, classifier outputs, and events over the network in a standardized way. Therefore, the TOBI CIP is not another BCI platform. In contrast, it facilitates distributed BCI research and interoperability between different BCI systems and platforms.

Design The design of the CIP is based on the BCI model proposed by Mason and Birch [28]. As shown in Figure 2, the CIP is based on a pipeline system. Data is acquired via a data acquisition system and forwarded to data processing modules. Different processing pipes are shown, because the TOBI CIP supports multiple (potentially distributed) processing streams. Modules are interconnected by different interfaces labeled as TiA, TiB, and TiC (TOBI interface A, B, and C). Each interface transmits specific types of signals used in BCI systems. A fourth interface (TiD) is used to transmit events and markers within the CIP. In case of multiple processing streams, a fusion module merges incoming information to one information stream. This merging process can be based on static or adaptive rules. The output of the fusion module can be used to control different types of applications or graphical user interfaces. The CIP synchronizes data streams by including the block number and time stamps of received data at each interface.

TiA TiA is an interface to transmit raw biosignals and information gathered from assistive devices or sensors [7]. Data is transmitted via TiA by the data acquisition and the preprocessing modules. TiA assigns different signal types to acquired data (for example, EEG, EOG, buttons, joystick, and so on) and supports simultaneous multi-rate and multi-channel data transmission. Furthermore, multiple clients can

[12] `www.tobi-project.org/download`

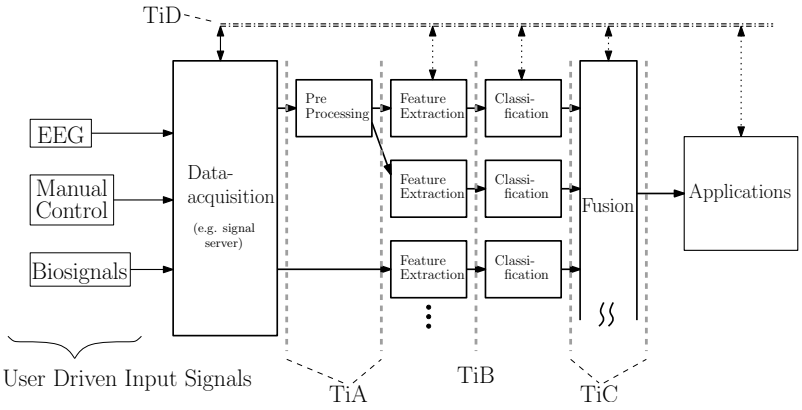

Fig. 2 Scheme of the TOBI Common Implementation Platform.

attach to a TiA server during runtime. Meta information is exchanged via a defined handshaking procedure based on XML (extensible markup language). Raw data can be transmitted either using TCP (transmission control protocol) or UDP using TiA data packets. A detailed documentation of TiA is available online[13].

TiB TiB is an interface for transmitting signal features such as band power. There is no further definition or implementation available yet.

TiC TiC is an interface to transmit detected classes and class labels within a BCI system. Information is encoded in XML messages. Each TiC message can consist of a different classifier and class, both with label and value fields. Therefore, the fusion module or an application module can interpret received TiC messages in a standardized way.

TiD is an interface to transmit markers and events used in BCI systems. It is based on XML messages and is acting like a bus system using multiple TCP connections. A module can send an event to the bus, and this event is dispatched by a TiD server (must be integrated or attached to the data acquisition system) to all connected clients.

Implementation A cross-platform library for TiA (implemented in C++) is available online[14,15]. Libraries for TiB, TiC, and TiD are currently under development and will be released soon. Additionally, a cross-platform data acquisition system called signal server (which implements TiA) is also available for download. The signal server supports simultaneous multi-rate data acquisition of different kinds of signals from different devices. The signal server and the TiA library have successfully passed various timing and stability tests. In addition, both software products are very resource and memory efficient, as they are implemented using C++. For

[13] `http://arxiv.org/abs/1103.4717v1`

[14] `www.tobi-project.org/download`

[15] `bci.tugraz.at/downloads`

cross-platform compatibility, only established libraries such as Boost[16] or SDL[17] are used within the TiA library or the signal server. TiA was already successfully integrated into MATLAB and Simulink, BCI2000, and a Linux embedded board (FOX Board G20, ARM 400 MHz, 64 MB RAM). MATLAB clients are currently available for TiA and TiC. Although there are no official builds for Mac OS X (or related platforms such as iOS) at the moment, the library can be built on these platforms. For example, we have successfully implemented an iOS app (running on iPhone, iPod Touch, and iPad) using the TOBI library. The integration of the TiA library into an embedded board or iOS-based devices demonstrates its portability and low resource requirements.

Benefits By using the TOBI Common Implementation Platform, it is possible to interconnect different BCI systems with a minimum of additional work. Since the CIP uses network connections, building distributed BCI systems is also straightforward. The signal server acquires data from different devices at the same time, potentially also with different sampling rates. Therefore, BCIs and other assistive technology can be combined into an augmented assistive device, the so-called hybrid BCI [34].

5 BCILAB

BCILAB[18] is an open-source MATLAB-based toolbox for advanced BCI research. Its graphical and scripting user interfaces provide access to a large collection of well-established methods, such as Common Spatial Patterns [40] and shrinkage LDA [6], as well as more recent developments [50, 18]. Because of its MATLAB foundation, the major strengths of the toolbox are implementing rapid prototyping, real time testing, offline performance evaluation of new BCI applications, and comparative evaluation of BCI methods. The design of BCILAB is less focused on clinical or commercial deployment, although compiled versions of BCILAB are available to run standalone versions of BCI methods.

Workflow Most BCI methods depend on parameters that may vary dramatically across people and/or sessions. These parameters must be learned, often via machine learning methods, on pre-recorded training or calibration data. Thus, building and using a BCI typically involves recording a calibration session, performing offline analyses on this data to learn or refine a BCI model, and using the learned model to estimate (in real time) changes in the user's cognitive state, response, or intent. Offline analysis in BCILAB involves computing models from training data, but frequently also extends to post-hoc/simulated assessment of the performance of a BCI model on separate testing data, thereby avoiding the need for costly online method testing sessions when sufficient data are available. To this end, BCILAB au-

[16] `www.boost.org`

[17] `www.libsdl.org`

[18] `sccn.ucsd.edu/wiki/BCILAB`

tomates rigorous cross-validation to assess test set performance, automatic parameter search, nested cross-validation, and online simulation. BCILAB also visualizes models, which facilitates psychophysiological interpretation of discriminating data features used by the model. For online processing, BCILAB provides a general-purpose real time data streaming and signal processing framework compatible with data collection and stimulus adaptation software (BCI2000, OpenViBE, ERICA), described below in more detail.

Features BCILAB puts emphasis on combining contemporary methods in machine learning, signal processing, statistical modeling, and electrophysiological imaging to facilitate methods-oriented research across disciplines. To this end, it provides several plug-in frameworks to speed up incorporation and testing of new BCI methods. Currently, BCILAB offers 15 machine learning methods, 20 signal processing methods (not counting variants), and 10 feature extraction methods, all of which can be configured and combined freely both via a GUI (as shown in Figure 3) and command line scripting. In addition to these dataflow-oriented components, BCI model building approaches can be realized that cut across several of these traditionally distinct processing stages, for example methods involving joint optimization and/or probabilistic modeling. To shorten the time it takes to realize a particular BCI approach, the toolbox makes heavy use of default settings when possible, and provides a pre-configured palette of well-established and recently-proposed BCI approaches, many of which can be reused with little customization.

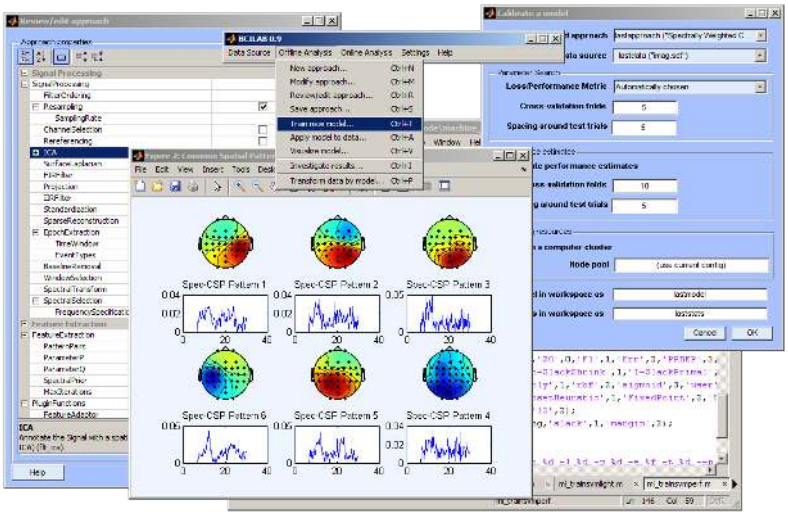

Fig. 3 The BCILAB graphical user interface showing the main menu (top middle), a model visualization window (bottom middle), a parameter settings dialog for a BCI approach (left), a method configuration window (right), as well as a MATLAB editor workspace (bottom right).

Through its linkage to EEGLAB [15], BCILAB makes available an extensive collection of neuroscience tools including the ability to operate on independent components found with Independent Component Analysis (ICA) methods [26], in particular Infomax [2] and Adaptive Mixture ICA (AMICA) [36]. Further capabilities include the use of prior information about relevant anatomical structures based on ICA-enabled source localization and probabilistic brain atlas look-up [21], and methods to extract high-quality source time-frequency representations including transient inter-source coherence. Furthermore, and unlike many current neuroscience workflows, these steps run fully automatically in most settings.

To support mobile brain/body imaging (MoBI) research [27], BCILAB has been designed to work with classifications based on multiple data modalities collected simultaneously, including EEG, eye gaze, body motion capture, and other biosignals, as recorded with the DataRiver framework in ERICA [16]. This feature may be especially relevant for applications of BCI methods outside the clinical context, in particular for passive monitoring of cognitive state in cognition-aware human-system interaction applications including gaming [61]. BCILAB uses plug-ins to link to real time recording and stimulation environments. Currently, it can either be used in standalone mode (with current support for BioSemi, TCP and OSC[19] data protocols) or as a signal processing module in a general-purpose BCI platform. Currently, BCI2000 [45], OpenViBE [41], and ERICA [16] are supported.

Availability BCILAB has been developed at the Swartz Center for Computational Neuroscience, University of California San Diego[20]. Its design was inspired by an earlier PhyPA toolbox developed by C. Kothe and T. Zander at Technical University, Berlin. BCILAB is open source (GPL) and supports most versions of MATLAB (running on Windows/Linux/Mac OS X).

6 BCI++

BCI++[21] is an open source framework based on a sophisticated graphics engine. The platform provides a set of tools for the rapid development of brain-computer interfaces and human-computer interaction (HCI) in general.

Structure of the system The BCI++ framework is composed of two main modules, which communicate with each other via TCP/IP. The first module is called HIM (Hardware Interface Module) and handles signal acquisition, storage, visualization, and real-time processing. The second module is named AEnima and provides a Graphical User Interface (GUI). This module is dedicated to creating and managing different protocols based on a high-level 2D/3D graphics engine. This structure was

[19] `opensoundcontrol.org`

[20] `sccn.ucsd.edu`

[21] `www.sensibilab.campuspoint.polimi.it`

devised to split the development of a real-time BCI system into two parts, namely into (1) signal processing algorithms and (2) a graphical user interface (GUI).

Hardware Interface Module (HIM) HIM provides a reliable software solution for the acquisition, storage, visualization, and real-time processing of signals. HIM communicates with AEnima via TCP/IP, but both software modules can also run on the same machine. HIM is open ource under the GNU GPL, the source code can be downloaded from the Sensibilab website or checked out from our Subversion repository (for the latest development version). HIM was written in C++ using the cross-platform wxWidgets library[22], but the actual release build is for Microsoft Windows only. HIM has a core block, which handles all tasks common to all protocols and loads plug-ins. These plug-ins are encapsulated in dynamically linked libraries and contain algorithms that the user develops. Algorithms for real-time signal processing can be designed both in C/C++ and MATLAB. BCI++ provides a Visual C++ 2010 project wizard to assist developers during the creation of new algorithm classes. The framework also comes with some SSVEP and motor imagery tools to help researchers rapidly create new BCI systems. In summary, BCI++ comes with a solid set of tools, which simplify the development of updates without rebuilding everything, and which allow to share applications and algorithms without recompiling them. Figure 4 (left) shows the main window, the signal plot window, and the feedback window of HIM, respectively.

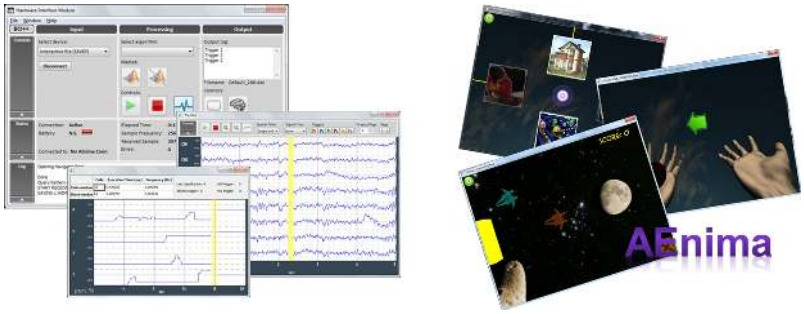

Fig. 4 Hardware Interface Module GUIs (left) and AEnima protocol examples (right).

HIM supports several signal acquisition devices; some are real, others are virtual and are useful for debugging and simulation purposes. The list of compatible devices includes: Brainproducts Brain Vision Recorder (supports most Brainproducts devices); Brainproducts Vamp; g.tec g.Mobilab; g.tec g.USBamp; Compumedics Neuroscan; Braintronics Brainbox (four different devices); SXT-telemed ProtheoII; and SXT-telemed Phedra.

BCI++ also provides compatibility with some of the devices developed in selected labs, including an Arduino-based amplifier that acquires up to 16 channels at 256 Hz (the electronic circuit and the Arduino sketch can be downloaded from the

[22] `www.wxwidgets.org`

BCI++ website). It is also possible to add a new device by deriving a specific class from the base instrument class. In the source code, an example illustrates how to implement new devices. More instructions are also reported in the documentation.

Graphical User Interface (AEnima) AEnima is a flexible tool developed to simplify the implementation of new operating protocols for BCI-based applications. There are two version of AEnima: one is written in C++ using a multiplatform graphics engine (Irrlicht[23]), whereas the other one is written in C# using XNA Game Studio to use BCI++ on Xbox 360, Windows Phone or Windows 7 Tablet platforms (the latter version is still under development). Both versions are open source and can be downloaded from the Sensibilab website or checked out from the Subversion repository.

The user interface software is based on a sophisticated graphics engine to provide a more realistic and challenging experience to the BCI user, and to guarantee versatility and efficiency in application development. Just like HIM, AEnima has a core based on these graphics engines and a plug-in which contains the real GUI. The two different versions (Irrlicht and XNA) both support OpenGL and DirectX (versions 8, 9, and 10). Therefore, the engine runs on fast and slow computers alike (for example, the software was successfully tested on an old Pentium 3 machine with an embedded graphics card). AEnima includes an audio engine, which offers a set of high-level functions which allow the reproduction and management of sound effects and audio files in different formats (for example WAV, MP3, and OGG). This engine also supports positional and 3D audio, which can be a useful way to develop protocols and paradigms with auditory stimulation or feedback. Furthermore, AEnima features two stimulation modules; the first one is a stimulation module that sends messages via USB to control external stimuli like the ones usually used for SSVEP BCI paradigms [37]. The second stimulation module can send commands via TCP/IP to a FES (functional electrical stimulation) controller used in BCIs for rehabilitation purposes. A specific software module was also implemented to provide an application layer with a home automation system. In the latest release, AEnima includes augmented reality features based on ARtoolkit[24]. Figure 4 (right) shows some AEnima GUI examples.

Conclusion The BCI++ system simplifies interfacing a BCI with external devices (such as a BCI-based FES stimulator for rehabilitation). The advanced graphics engine allows developers to focus on the design of the HCI aspect without having to spend a long time on developing a new system from scratch. BCI++ supports different kinds of acquisition devices, which could be used by both the end-user in their daily activities (for example, home automation control) and by the researcher to develop new protocols, algorithms, and software modules useful in a BCI laboratory. The framework is very flexible, and the large set of debugging tools dramatically simplifies debugging and testing of a new system.

---

[23] `irrlicht.sourceforge.net`

[24] `www.hitl.washington.edu/artoolkit`

However, the most relevant aspect of BCI++ is the possibility for unskilled developers to develop and test their own work and to actively help to increase the number of available instruments in the framework. All software modules and the source code are available on our web site along with some examples and documentation. The framework was also validated and tested on more than one-hundred users (healthy and disabled) with SSVEP and motor imagery BCI systems.

7 xBCI

xBCI[25] is a generic platform for developing online brain-computer interfaces [49]. This platform provides users with an easy-to-use system development tool and reduces the time needed to develop a BCI system. The main features of this platform are as follows:

- Extendable and modular system design: Functional modules can be added by users, and PCs or data acquisition devices (e. g. EEG or NIRS amplifiers) can be easily integrated into xBCI.
- GUI-based system development: A GUI-based editor for building and editing the BCI system is provided. Using the editor, even inexperienced users can easily build their own systems.
- Multi-threaded parallel processing: Users can build a multi-threaded parallel processing system without any detailed knowledge of the operating system or thread programming.
- Multi-OS support: The platform supports multiple operating systems, such as Microsoft Windows and GNU Linux.
- Open source: The xBCI platform was implemented with the GNU C/C++ compiler set, and only open source libraries were used to implement components and the platform itself. It does not depend on any commercial software products.

The platform consists of several functional modules (components), which can be used to realize a specific BCI system. Users can design and build various types of BCI systems by combining these components. The ready-to-use components are listed below.

- Basic mathematical operations: Logical operation, arithmetic operation of scalar values and matrices, and basic mathematical functions such as trigonometric and logarithmic functions. Mathematical expressions are evaluated and calculated by these dedicated components.
- Data processing: Temporal and spatial filters, frequency analysis, averaging, pattern classifiers, data import and export, and so on.
- Data acquisition: Measured data or digital event marker signals are acquired by interface boards (e. g. A/D converter boards) or parallel ports.

[25] `xbci.sourceforge.net`

- Network communications: Data transfer from/to other PCs or data acquisition devices over TCP/IP or UDP. These components allow users to easily build an experimental system with several PCs or data acquisition devices which are connected over a network.
- Data visualization: Real time data scopes for displaying and monitoring measured or processed data.
- Experiment control: Control of experimental protocols with a precise timing accuracy.
- Real time feedback presentation: Various ways to present the feedback information for neurofeedback experiments can be constructed.

Users can also add custom components to extend the functionality of the platform. A custom component can be added to the platform by either programming in C++ or by using a scripting language. Every component is completely independent as a plug-in, and components can be added or modified without rebuilding the whole platform. Plug-ins can then be distributed separately from the platform.

Each component is executed in its own thread and starts processing in parallel as soon as any incoming data becomes available. Data are transferred between components by means of a packet. A packet consists of a packet header and data to be processed. System parameters, such as sampling frequency and number of channels for measurement, are shared among components by the packet header.

Figures 5 and 6 show the application of xBCI to the online BCI neurofeedback training system based on a brain switch [2, 3], which detects a binary command (on/off) by an increase of EEG band power elicited during motor imagery recorded from a single bipolar EEG channel. Figure 5 shows the block diagram of the data processing. The data acquisition, online processing, and neurofeedback experiment control were carried out on PC-I, and the measured data were transmitted to PC-II and displayed for online monitoring. The realized system by xBCI is shown in Figure 6. This data processing chain was implemented by connecting the components in the GUI editor (upper left), and the recorded EEG data (middle), the spectrum (lower left), as well as neurofeedback information (right) were displayed.

In summary, the xBCI platform provides users with an easy-to-use system development tool and reduces the time needed to develop a BCI system. The complete platform along with documentation and example designs can be obtained from the project website and is freely available under the GNU General Public License.

8 BF++

The aim of BF++[26] (Body Language Framework in C++) is to provide tools for the implementation, modeling and data analysis of BCI and HCI systems. The main objective of BF++ is to create unique methods, terminologies, and tools independent from the specific protocols such as P300, SSVEP, or SMR BCIs. BF++ is based

[26] www.brainterface.com

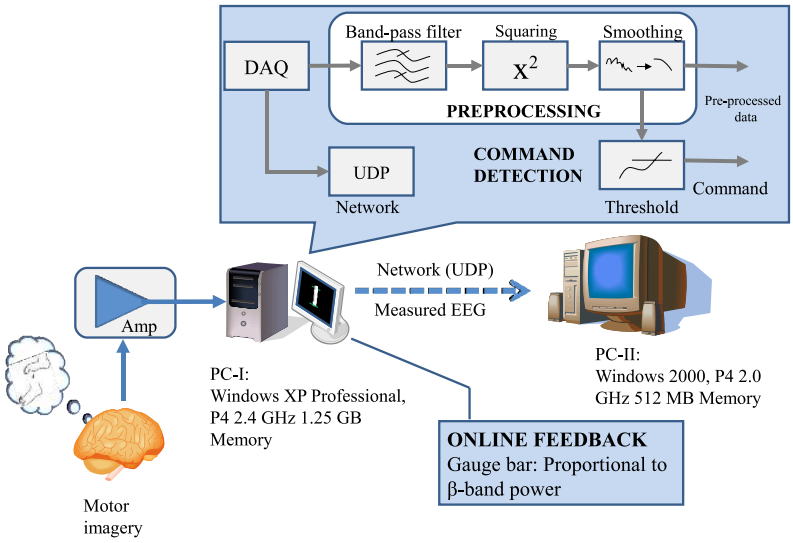

Fig. 5 Schematic diagram of the data processing chain in an example neurofeedback application.

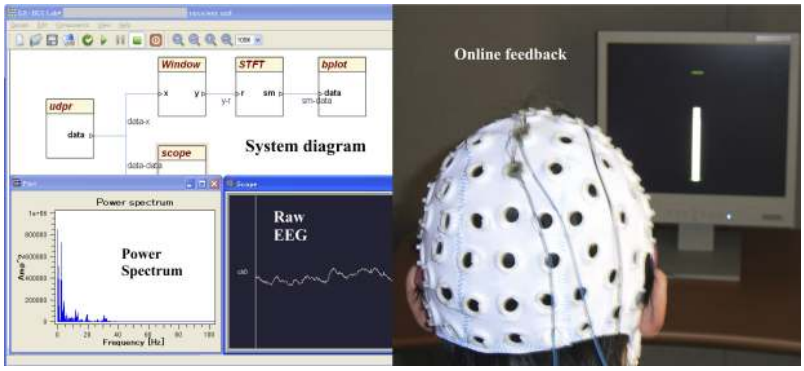

Fig. 6 The example neurofeedback application.

on a well-defined abstract model, on top of which various methods and tools have been implemented. It is highly scalable, cross-platform, and programmed by adopting only well established technologies, such as C++ as the programming language, XML for storage, and UML (unified modeling language) for description and documentation. Great effort has been made to allow a reliable comparison among different systems and the optimization of their performances. This was achieved by starting from a unique static functional model [28] as shown in Figure 7. In this model, the two main elements are the transducer, which is responsible for the acquisition of neurophysiological signals and their classification, and the control interface, which processes the output of the classifier and controls external peripheral devices by feeding into the application control module.

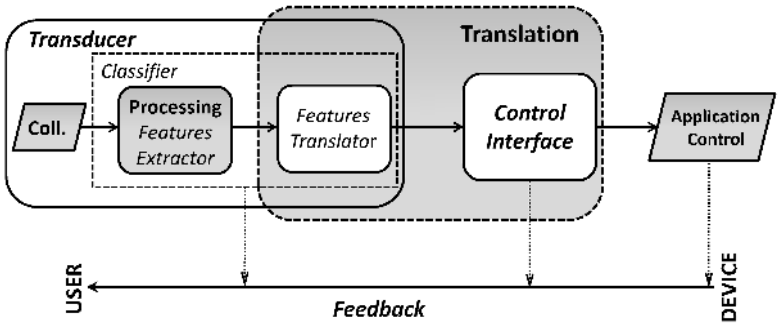

Fig. 7 Functional model of a BCI system as used in BF++.

This model was extended recently by adding dynamic behavior and a description of the model using UML sequence diagrams [39]. Following this model, the same actors (classes in object oriented programming terminology) have been successfully used in five different BCI protocols confirming its robustness and the high abstraction level achieved.

Another important aspect of BF++ is that it provides a unique and reliable performance metric, the efficiency [4] of the model. It is based on the characterization of either the transducer or the control interface and it is able to deal with their adaptation. Compared to other commonly used metrics (such as classification accuracy, information transfer rate, and so on), the efficiency is suitable for the description, simulation, and more importantly, optimization of the systems. For this reason, several software tools (the BF++ Toys) have been released. The advantage of using the same model and methods regardless of the specific protocol maximizes consistency among the tools and their (re)usability.

Moreover, specific file formats have been implemented using XML, which allows extensions by adding data without breaking the backward compatibility with already existing tools. To facilitate the exchange of data between different laboratories, support for several file formats are provided (for example, BCI2000, GDF, Brain Vision Analyzer, EDF, CTF MEG, and so on). However, only the BF++ native NPX file format (neurophysiological data in XML) is able to take advantage of all BF++ software analysis packages [5]. These packages allow to perform EEG and ERP analysis, spectral analysis, statistics, spatial filtering (for example, independent component analysis and common spatial patterns), classification, and 2D/3D mapping. All packages can be downloaded from the project website.

9 Pyff

Pyff[27] (Pythonic feedback framework) is a framework for the rapid development of experimental paradigms and a platform to run neuroscientific experiments. The foremost design goal was to make the development of BCI feedback and stimulus applications as fast and easy as possible. As stimulation and feedback paradigms are getting more and more ambitious and complex, one bottleneck in the process of conducting experiments becomes the actual development of the software. This problem is even more severe in labs where such software is not developed by computer scientists. Thus, we decided to implement the framework in Python. Python is a high level programming language and well known for its flat learning curve compared to low level languages like C or C++. Experience has shown us that non-expert programmers typically learn to program feedback and stimulus applications with Pyff within two days. Implementing equivalent applications in low level programming languages like C or C++ can easily take an order of magnitude more time, and even for experienced programmers usually a factor of two remains [38].

Pyff is completely written in Python and thus not tied to a special operating system. Pyff runs everywhere where Python runs, which in includes all major platforms such as Linux, Mac OS X, and Windows. Moreover, we tried our best to keep Pyff also independent from specific BCI systems. That is, our goal was to make it compatible with as many BCI systems as possible. We achieved that by coupling Pyff with the rest of the BCI system using UPD and XML. The network protocol is used to transport the data from the BCI system to Pyff, and XML is used to wrap arbitrary data in a format Pyff can handle. UDP is supported by almost all mainstream programming languages, and so is XML. A complete description of the interface can be found in [53]. Additionally, Pyff also supports the TOBI interface to communicate with the rest of the BCI system.

It is important to note that Pyff does not provide a complete BCI software stack. In a typical BCI environment, a BCI system consists of three parts: (1) data acquisition, (2) signal processing, and (3) feedback or stimulus presentation. Pyff provides only the third part of this stack. Moreover, it creates a layer above the BCI system and allows to implement feedback and stimuli without having to worry about the underlying BCI system. Therefore, Pyff is not only a framework for rapid development of feedback and stimulus applications, but also a platform to run neuroscientific experiments independent from BCI systems. Such a platform could foster a fruitful exchange of experimental paradigms between research groups, decrease the need of reprogramming standard paradigms, facilitate the reproducibility of published results, and promote standardization of feedback and stimulus presentation.

Pyff already comes with a variety of ready-to-use experimental paradigms, like the hex-o-speller or the matrix speller. Pyff is actively maintained by one developer and several others are regularly contributing code.

---

[27] `bbci.de/pyff`

Overview of Pyff Pyff consists of four parts: (1) the feedback controller, (2) a graphical user interface, (3) a set of feedback paradigms and stimuli, and (4) a collection of base classes.

The feedback controller receives incoming signals from the BCI system and translates and forwards them to the feedback and stimulus application. The feedback controller is also responsible for controlling the execution of these applications, for example starting, pausing or stopping them.

The graphical user interface (GUI) controls the feedback controller remotely over the network. The experimenter can select, start, pause, and stop feedback and stimulus applications as well as inspect and modify their variables during runtime. Being able to modify all variables on the fly provides a great way to explore different settings in a pilot experiment, and this feature also makes the GUI an invaluable debugging tool. The GUI communicates with the feedback controller using the same UDP/XML protocol as the BCI system. This makes the GUI completely optional, every command can also be issued by the BCI system directly.

Pyff also provides a constantly growing set of ready-to-go feedback and stimulus applications, which can be used without or with only small modifications. Pyff supports loading and saving the feedback and stimulus application's parameters to a JSON[28] file, which is useful for providing supporting material in publications and facilitates the reproducibility of results.

The collection of feedback base classes provides methods and functionality shared by many feedback and stimulus applications. These methods can be used in derived classes, which reduces the overhead of developing new applications and minimizes code duplication. For example, Pygame[29] is often used for the graphical representation of stimuli. Applications using Pygame often share a huge amount code, for example for the initialization of the screen or polling Pygame's event queue. All this functionality is already available in the `PygameFeedback` base class and does not have to be rewritten in derived classes. All feedback base classes also provide the methods needed to communicate with the feedback controller. Therefore, every class derived from one of the feedback base classes is automatically a valid feedback (or stimulus) class.

Since Python can utilize existing libraries (e. g. shared objects or DLLs), it is straightforward to use special hardware within Pyff. Pyff already provides support for the IntelliGaze eye tracker by Alea Technologies and the g.STIMbox by g.tec.

License and Availability Pyff is completely open source and free software under the terms of the GNU General Public License. Pyff is available for download including documentation as well as other information and links on the project homepage. Furthermore, the source code is available from the public Git repository. The requirements to run Pyff are currently a working installation of Python 2.6.6[30] and PyQt 4[31].

[28] `www.json.org`

[29] `www.pygame.org`

[30] `www.python.org`

[31] `www.riverbankcomputing.co.uk/software/pyqt/intro`

10 Summary and Conclusion

The number of user-friendly BCI software platforms has increased significantly over the past years. The days when a researcher had to start from scratch and develop all required BCI components are almost over, or at least there are viable alternatives available. Nowadays, people who want to use or develop BCIs can choose between many publicly available BCI platforms. We have described seven of the most popular BCI frameworks and one platform dedicated to feedback and stimulus presentation in this article. While some platforms have been available for many years and offer a great number of features (for example, BCI2000 and OpenViBE), each platform has its unique features and benefits. We addressed topics that might be important to potential users, such as licensing issues, availability for multiple platforms, supported hardware devices, interaction with other software applications, and so on. Table 1 compares all platforms with respect to supported operating systems, license, and requirements (see caption for more details).

Table 1 Feature comparison of BCI platforms. Columns 2–4 indicate if operating systems are officially supported. Support for Windows includes versions XP, Vista, and 7 unless otherwise noted. Support for Mac OS X includes versions 10.5 and 10.6 unless otherwise noted. If Linux is supported, the platform should run on any Linux distribution. The last column lists all required software components that are not open source or freely available.

Platform	Windows	Mac OS X	Linux	License	Requirements
BCI2000	•	–[a]	–[a]	GPL	Windows[b]
OpenViBE	•[c]	–	•	LGPL[d]	–
TOBI	•	–[e]	•	GPL, LGPL[f]	–
BCILAB	•[g]	•[g]	•[g]	GPL	MATLAB[h]
BCI++	•	–[i]	–[i]	GPL	Windows[i]
xBCI	•	•	•	GPL	–
BF++	•[j]	–[i]	–[i]	Free[k]	Windows[i]
Pyff	•[l]	•[l]	•[l]	GPL	–

[a] Officially supported in next version, current version should run under Mac OS X and Linux.
[b] Next version will also run under Mac OS X and Linux.
[c] Also runs on Windows 2000.
[d] Version 2 or later.
[e] Unofficially, the TOBI library runs on Mac OS X and iOS platforms.
[f] TiA is licensed under the LGPL; the TOBI Signal Server is licensed under the GPL.
[g] All versions that run MATLAB R2006a or greater.
[h] MATLAB is not required to run BCILAB, only for making changes at the source code level.
[i] Unofficially, also runs and compiles on Mac OS X and Linux.
[j] Unofficially, BF++ also compiles on Windows CE.
[k] Free for non-commercial use.
[l] All versions that run Python 2.6.6.

Future directions could include exploiting synergies and minimizing redundancies between platforms. The TOBI CIP could play an important role in reaching this goal, or in reaching less ambitious short term goals such as making the platforms

talk to another. This would allow the data acquisition facility from one platform to be used with the feature extraction facility of another platform and the visualization capabilities of a third framework. It should be relatively straightforward to adapt existing platforms to support the TOBI interfaces in addition to their native data exchange formats. Even if platform-specific features had to be dropped because of a lack of support in the TOBI protocols, the possibility to use this standardized format opens up a wealth of opportunities. Work towards implementing TOBI interfaces (especially TiA) has already started in some platforms, and is planned for other frameworks. For example, Pyff has had built-in support for the TOBI interfaces for several months.

In summary, there probably is no best platform for everyone. With the information presented in this article, interested users should be able to identify platforms that might be suitable for their specific purposes.

Acknowledgements The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the corresponding funding agencies. The authors would like to acknowledge the following projects and funding sources:

BCI2000: This work was supported by grants from the US Army Research Office (W911NF-07-1-0415, W911NF-08-1-0216) and the NIH/NIBIB (EB006356 and EB000856).

OpenViBE: This work was partly supported by grants of the French National Research Agency under the OpenViBE (ANR-05-RNTL-016) and OpenViBE2 (ANR-09-CORD-017) projects.

TOBI: This work is supported by the European ICT Programme Project FP7-224631.

BCILAB: Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0022. Initial development was supported by a gift from the Swartz Foundation (Oldfield, NY) and a basic research grant of the Office of Naval Research (ONR).

Pyff: This work was partly supported by grants of the Bundesministerium für Bildung und Forschung (BMBF) (FKZ 01IB001A, 01GQ0850) and by the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886.

References

1. Allison, B.Z., McFarland, D.J., Schalk, G., Zheng, S.D., Jackson, M.M., Wolpaw, J.R.: Towards an independent brain-computer interface using steady state visual evoked potentials. Clinical Neurophysiology **119**, 399–408 (2008)
2. Bell, A.J., Sejnowski, T.J.: An information-maximization approach to blind separation and blind deconvolution. Neural Computation **7**, 1129–1159 (1995)
3. Bell, C.J., Shenoy, P., Chalodhorn, R., Rao, R.P.: Control of a humanoid robot by a noninvasive brain-computer interface in humans. Journal of Neural Engineering **5**, 214–220 (2008)
4. Bianchi, L., Quitadamo, L., Garreffa, G., Cardarilli, G., Marciani, M.: Performances evaluation and optimization of brain computer interface systems in a copy spelling task. IEEE Transactions on Neural Systems and Rehabilitation Engineering **15**, 207–216 (2007)
5. Bianchi, L., Quitadamo, L.R., Abbafati, M., Marciani, M.G., Saggio, G.: Introducing NPXLab 2010: a tool for the analysis and optimization of P300 based brain-computer interfaces. In: 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies, pp. 1–4 (2009)

6. Blankertz, B., Lemm, S., Treder, M., Haufe, S., Müller, K.R.: Single-trial analysis and classification of ERP components – a tutorial. NeuroImage **56**, 814–825 (2011)
7. Breitwieser, C., Neuper, C., Müller-Putz, G.R.: A concept to standardize raw biosignal transmission for brain-computer interfaces. In: Proceedings of the 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2011)
8. Brouwer, A.M., Van Erp, J.B.F.: A tactile P300 brain-computer interface. Frontiers in Neuroscience **4** (2010)
9. Brunner, P., Ritaccio, A.L., Emrich, J.F., Bischof, H., Schalk, G.: Rapid communication with a "P300" matrix speller using electrocorticographic signals (ECoG). Frontiers in Neuroscience **5** (2011)
10. Brunner, P., Ritaccio, A.L., Lynch, T.M., Emrich, J.F., Wilson, J.A., Williams, J.C., Aarnoutse, E.J., Ramsey, N.F., Leuthardt, E.C., Bischof, H., Schalk, G.: A practical procedure for real-time functional mapping of eloquent cortex using electrocorticographic signals in humans. Epilepsy & Behavior **15**, 278–286 (2009)
11. Buch, E., Weber, C., Cohen, L.G., Braun, C., Dimyan, M.A., Ard, T., Mellinger, J., Caria, A., Soekadar, S., Fourkas, A., Birbaumer, N.: Think to move: a neuromagnetic brain-computer interface (BCI) system for chronic stroke. Stroke **39**, 910–917 (2008)
12. Cabrera, A.F., Dremstrup, K.: Auditory and spatial navigation imagery in brain-computer interface using optimized wavelets. Journal of Neuroscience Methods **174**, 135–146 (2008)
13. Cincotti, F., Mattia, D., Aloise, F., Bufalari, S., Astolfi, L., De Vico Fallani, F., Tocci, A., Bianchi, L., Marciani, M.G., Gao, S., Millán, J., Babiloni, F.: High-resolution EEG techniques for brain-computer interface applications. Journal of Neuroscience Methods **167**, 31–42 (2008)
14. Cincotti, F., Mattia, D., Aloise, F., Bufalari, S., Schalk, G., Oriolo, G., Cherubini, A., Marciani, M.G., Babiloni, F.: Non-invasive brain-computer interface system: towards its application as assistive technology. Brain Research Bulletin **75**, 796–803 (2008)
15. Delorme, A., Makeig, S.: EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. Journal of Neuroscience Methods **134**, 9–21 (2004)
16. Delorme, A., Mullen, T., Kothe, C., Acar, Z.A., Bigdely-Shamlo, N., Vankov, A., Makeig, S.: EEGLAB, SIFT, NFT, BCILAB, and ERICA: new tools for advanced EEG processing. Computational Intelligence and Neuroscience **2011**, 130,714 (2011)
17. Felton, E.A., Wilson, J.A., Williams, J.C., Garell, P.C.: Electrocorticographically controlled brain-computer interfaces using motor and sensory imagery in patients with temporary subdural electrode implants – report of four cases. Journal of Neurosurgery **106**, 495–500 (2007)
18. Kothe, C., Makeig, S.: Estimation of task workload from EEG data: new and current tools and perspectives. In: Proceedings of the 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2011)
19. Kubánek, J., Miller, K.J., Ojemann, J.G., Wolpaw, J.R., Schalk, G.: Decoding flexion of individual fingers using electrocorticographic signals in humans. Journal of Neural Engineering **6**, 066,001 (2009)
20. Kübler, A., Nijboer, F., Mellinger, J., Vaughan, T.M., Pawelzik, H., Schalk, G., McFarland, D.J., Birbaumer, N., Wolpaw, J.R.: Patients with ALS can use sensorimotor rhythms to operate a brain-computer interface. Neurology **64**, 1775–1777 (2005)
21. Lancaster, J.L., Woldorff, M.G., Parsons, L.M., Liotti, M., Freitas, C.S., Rainey, L., Kochunov, P.V., Nickerson, D., Mikiten, S.A., Fox, P.T.: Automated Talairach atlas labels for functional brain mapping. Human Brain Mapping **10**, 120–131 (2000)
22. Leuthardt, E.C., Miller, K.J., Anderson, N.R., Schalk, G., Dowling, J., Miller, J., Moran, D.W., Ojemann, J.G.: Electrocorticographic frequency alteration mapping: a clinical technique for mapping the motor cortex. Neurosurgery **60**, 260–270 (2007)
23. Leuthardt, E.C., Miller, K.J., Schalk, G., Rao, R.P., Ojemann, J.G.: Electrocorticography-based brain computer interface – the Seattle experience. IEEE Transactions on Neural Systems and Rehabilitation Engineering **14**, 194–198 (2006)

24. Leuthardt, E.C., Schalk, G., Wolpaw, J.R., Ojemann, J.G., Moran, D.W.: A brain-computer interface using electrocorticographic signals in humans. Journal of Neural Engineering **1**, 63–71 (2004)
25. Lotte, F., Renard, Y., Lécuyer, A.: Self-paced brain-computer interaction with virtual worlds: a qualitative and quantitative study 'out-of-the-lab'. In: Proceedings of the Fourth International Brain-Computer Interface Workshop and Training Course, pp. 373–378 (2008)
26. Makeig, S., Bell, A.J., Jung, T.P., Sejnowski, T.J.: Independent component analysis of electroencephalographic data. In: D. Touretzky, M. Mozer, M. Hasselmo (eds.) Advances in Neural Information Processing Systems, pp. 145–151. MIT Press (1996)
27. Makeig, S., Gramann, K., Jung, T.P., Sejnowski, T.J., Polzner, H.: Linking brain, mind and behavior. International Journal of Psychophysiology **73**, 95–100 (2009)
28. Mason, S.G., Birch, G.E.: A general framework for brain-computer interface design. IEEE Transactions on Neural Systems and Rehabilitation Engineering **11**, 70–85 (2003)
29. McFarland, D.J., Krusienski, D.J., Sarnacki, W.A., Wolpaw, J.R.: Emulation of computer mouse control with a noninvasive brain-computer interface. Journal of Neural Engineering **5**, 101–110 (2008)
30. McFarland, D.J., Sarnacki, W.A., Wolpaw, J.R.: Electroencephalographic (EEG) control of three-dimensional movement. Journal of Neural Engineering **7**, 036,007 (2010)
31. Mellinger, J., Schalk, G., Braun, C., Preissl, H., Rosenstiel, W., Birbaumer, N., Kübler, A.: An MEG-based brain-computer interface (BCI). NeuroImage **36**, 581–593 (2007)
32. Miller, K.J., Dennijs, M., Shenoy, P., Miller, J.W., Rao, R.P., Ojemann, J.G.: Real-time functional brain mapping using electrocorticography. Neuroimage **37**, 504–507 (2007)
33. Miller, K.J., Leuthardt, E.C., Schalk, G., Rao, R.P., Anderson, N.R., Moran, D.W., Miller, J.W., Ojemann, J.G.: Spectral changes in cortical surface potentials during motor movement. Journal of Neuroscience **27**, 2424–32 (2007)
34. Millán, J., Rupp, R., Müller-Putz, G.R., Murray-Smith, R., Giugliemma, C., Tangermann, M., Vidaurre, C., Cincotti, F., Kübler, A., Leeb, R., Neuper, C., Müller, K.R., Mattia, D.: Combining brain-computer interfaces and assistive technologies: state-of-the-art and challenges. Frontiers in Neuroscience **4** (2010)
35. Müller-Putz, G.R., Kaiser, V., Solis-Escalante, T., Pfurtscheller, G.: Fast set-up asynchronous brain-switch based on detection of foot motor imagery in 1-channel EEG. Medical and Biological Engineering and Computing **48**, 229–233 (2010)
36. Palmer, J.A., Makeig, S., Kreutz-Delgado, K., Rao, B.D.: Newton Method for the ICA Mixture Model. In: Proceedings of the 33rd IEEE International Conference on Acoustics and Signal Processing (ICASSP), pp. 1805–1808 (2008)
37. Parini, S., Maggi, L., Turconi, A.C., Andreoni, G.: A robust and self-paced BCI system based on a four class SSVEP paradigm: algorithms and protocols for a high-transfer-rate direct brain communication. Computational Intelligence and Neuroscience **2009**, 864,564 (2009)
38. Prechelt, L.: An empirical comparison of seven programming languages. IEEE Computer **33**, 23–29 (2000)
39. Quitadamo, L.R., Marciani, M.G., Cardarilli, G.C., Bianchi, L.: Describing different brain computer interface systems through a unique model: a UML implementation. Neuroinformatics **6**, 81–96 (2008)
40. Ramoser, H., Müller-Gerking, J., Pfurtscheller, G.: Optimal spatial filtering of single trial EEG during imagined hand movement. IEEE Transactions on Rehabilitation Engineering **8**, 441–446 (2000)
41. Renard, Y., Lotte, F., Gibert, G., Congedo, M., Maby, E., Delannoy, V., Bertrand, O., Lécuyer, A.: OpenViBE: an open-source software platform to design, test, and use brain–computer interfaces in real and virtual environments. Presence **19**, 35–53 (2010)
42. Royer, A.S., He, B.: Goal selection versus process control in a brain-computer interface based on sensorimotor rhythms. Journal of Neural Engineering **6**, 016,005 (2009)
43. Schalk, G., Kubánek, J., Miller, K.J., Anderson, N.R., Leuthardt, E.C., Ojemann, J.G., Limbrick, D., Moran, D., Gerhardt, L.A., Wolpaw, J.R.: Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. Journal of Neural Engineering **4**, 264–275 (2007)

44. Schalk, G., Leuthardt, E.C., Brunner, P., Ojemann, J.G., Gerhardt, L.A., Wolpaw, J.R.: Real-time detection of event-related brain activity. NeuroImage **43**, 245–249 (2008)
45. Schalk, G., McFarland, D.J., Hinterberger, T., Birbaumer, N., Wolpaw, J.R.: BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. IEEE Transactions on Biomedical Engineering **51**, 1034–1043 (2004)
46. Schalk, G., Mellinger, J.: A Practical Guide to Brain-Computer Interfacing with BCI2000: General-Purpose Software for Brain-Computer Interface Research, Data Acquisition, Stimulus Presentation, and Brain Monitoring. Springer (2010)
47. Schalk, G., Miller, K.J., Anderson, N.R., Wilson, J.A., Smyth, M.D., Ojemann, J.G., Moran, D.W., Wolpaw, J.R., Leuthardt, E.C.: Two-dimensional movement control using electrocorticographic signals in humans. Journal of Neural Engineering **5**, 75–84 (2008)
48. Sellers, E.W., Vaughan, T.M., Wolpaw, J.R.: A brain-computer interface for long-term independent home use. Amyotrophic Lateral Sclerosis **11**, 449–455 (2010)
49. Susila, I.P., Kanoh, S., Miyamoto, K., Yoshinobu, T.: xBCI: a generic platform for development of an online BCI system. IEEJ Transactions on Electrical and Electronic Engineering **5**, 467–473 (2010)
50. Tomioka, R., Müller, K.R.: A regularized discriminative framework for EEG analysis with application to brain–computer interface. NeuroImage **49**, 415–432 (2010)
51. Valderrama, A.T., Oostenveld, R., Vansteensel, M.J., Huiskamp, G.M., Ramsey, N.F.: Gain of the human dura in vivo and its effect on invasive brain signals feature detection. Journal of Neuroscience Methods **187**, 270–279 (2010)
52. Vaughan, T.M., McFarland, D.J., Schalk, G., Sarnacki, W.A., Krusienski, D.J., Sellers, E.W., Wolpaw, J.R.: The Wadsworth BCI Research and Development Program: at home with BCI. IEEE Transactions on Neural Systems and Rehabilitation Engineering **14**, 229–233 (2006)
53. Venthur, B., Scholler, S., Williamson, J., Dähne, S., Treder, M.S., Kramarek, M.T., Müller, K.R., Blankertz, B.: Pyff – a pythonic framework for feedback applications and stimulus presentation in neuroscience. Frontiers in Neuroscience **4** (2010)
54. Vidal, J.J.: Toward direct brain-computer communication. Annual Review of Biophysics and Bioengineering **2**, 157–180 (1973)
55. Wilson, J.A., Felton, E.A., Garell, P.C., Schalk, G., Williams, J.C.: ECoG factors underlying multimodal control of a brain-computer interface. IEEE Transactions on Neural Systems and Rehabilitation Engineering **14**, 246–250 (2006)
56. Wilson, J.A., Mellinger, J., Schalk, G., Williams, J.: A Procedure for Measuring Latencies in Brain-Computer Interfaces. IEEE Transactions on Biomedical Engineering **7**, 1785–1797 (2010)
57. Wisneski, K.J., Anderson, N., Schalk, G., Smyth, M., Moran, D., Leuthardt, E.C.: Unique cortical physiology associated with ipsilateral hand movements and neuroprosthetic implications. Stroke **39**, 3351–3359 (2008)
58. Wolpaw, J.R., McFarland, D.J.: Multichannel EEG-based brain-computer communication. Clinical Neurophysiology **90**, 444–449 (1994)
59. Wolpaw, J.R., McFarland, D.J.: Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. Proceedings of the National Academy of Sciences USA **101**, 17,849–17,854 (2004)
60. Yamawaki, N., Wilke, C., Liu, Z., He, B.: An enhanced time-frequency-spatial approach for motor imagery classification. IEEE Transactions on Neural Systems and Rehabilitation Engineering **14**, 250–254 (2006)
61. Zander, T.O., Kothe, C.: Towards passive brain-computer interfaces: applying brain-computer interface technology to human-machine systems in general. Journal of Neural Engineering **8**, 025,005 (2011)