Brain–Computer Interface Software: A Review and Discussion

Pierce Stegman, Chris S. Crawford, Marvin Andujar, Anton Nijholt, and Juan E. Gilbert, *Senior Member, IEEE*

***Abstract*—Software is a critical component of brain–computer interfaces (BCIs). While BCI hardware enables the retrieval of brain signals, BCI software is required to analyze these signals, produce output, and provide feedback. Users from multiple research areas have adopted BCI software platforms to investigate various concepts. Recently, interest in web-based BCI software has also emerged. The system design and control signal techniques of state-of-the-art BCI software platforms have been previously investigated. However, there is limited literature discussing user adoption of BCI software platforms. Additionally, there is a lack of work discussing the recent emergence of web tools relevant to BCI applications. This article aims to address these gaps by presenting a bibliometric review of the state-of-the-art BCI software. Furthermore, we discuss web-based BCIs and present tools that may be used to develop future web-based BCI applications.**

***Index Terms*—Bibliometric, brain–computer interface (BCI), platforms, signal processing, software.**

I. Introduction

BRAIN–COMPUTER interface (BCI) systems convert central nervous system (CNS) activity to artificial output that is then used to replace, restore, enhance, supplement, or improve natural CNS output [1]. BCI algorithms identify patterns in brain signals and take actions based on the observed patterns. This process allows users to interact with their environment without having to use their peripheral nerves and muscles. Electrical signals from the brain were first captured from the cortical surface of animals in 1875 [2]. In 1929, Berger [3] reported the first successful attempt to capture brain signals from the human scalp. Despite this significant accomplishment, the technology necessary to efficiently measure and process brain signals remained limited during this era. As technology advanced during the 20th century, precise measurements of electrical activity became more feasible. In 1973, Vidal [4], [5] coined the term "BCI" after developing a computer-based system that converted electrical signals captured from the scalp to commands. BCI research experienced gradual progress between the mid 1970s and late 1990s. During this time, many research labs developed their own custom BCI systems. Unfortunately, this resulted in a lack of standardization and limited opportunities to replicate and compare published studies [6].

During the early 2000s, new BCI tools began to appear [8]–[10]. Since the introduction of general-purpose BCI systems, several open-source BCI platforms have been released [11]. These platforms have predominantly been used to develop BCI technology that falls into the following categories: basic research, clinical/translational research, consumer products, and emerging applications [12]. Clinical/translational applications often provide assistive care for people with clinical conditions. Examples of these BCI applications include BCI controlled wheelchairs [13]–[15], prosthetic devices [16], and virtual keyboards [17]. Along with aiding communication and control, BCIs are also used for motor recovery training during rehabilitation therapy [18]. Recent BCI research has resulted in nonmedical applications [19]. Applications deriving from the nonmedical domain usually fall into the consumer products or emerging application categories. Nonmedical BCI applications include drowsiness detection while driving [20], workload estimation [21], [22], gaming [23], engagement monitoring [24], neurofeedback training [25], and neuromarketing [26]. Previous reviews have discussed the system design and control signal techniques of earlier BCI tools [11], [27].

However, there is a lack of work examining adoption patterns (i.e., research areas) of BCI software platforms. Furthermore, there is limited research discussing the recent emergence of web-based BCI systems. This article aims to address this gap by presenting a bibliometric review of BCI software platforms discussed in previous reviews. By utilizing this approach, we intend to equip readers with objective insights on the use of BCI software platforms since the early 2000s. Additionally, we discuss web-based BCI systems and present tools that may assist future BCI researchers.

The rest of the article is organized as follows. Section II provides readers with information regarding the background and adoption of earlier BCI software platforms. This section does not focus on system design and control signal aspects of BCI software platforms as previous articles have discussed this topic [11], [27]. Afterward, previously reported BCI software

Manuscript received March 2, 2019; revised October 28, 2019; accepted December 14, 2019. Date of publication February 12, 2020; date of current version March 12, 2020. This article was recommended by Associate Editor R. Chavarriaga. *(Corresponding author: Chris S. Crawford.)*

P. Stegman and C. S. Crawford are with the Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487-0166 USA (e-mail: pwstegman@crimson.ua.edu; crawford@cs.ua.edu).

M. Andujar is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: andujar1@usf.edu).

A. Nijholt is with the Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7522 Enschede, The Netherlands (e-mail: a.nijholt@utwente.nl).

J. E. Gilbert is with the Department of Computer and Information Science and Engineering, University of Florida, Tuscaloosa, AL 35403-2454 USA (e-mail: juan@ufl.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2020.2968411

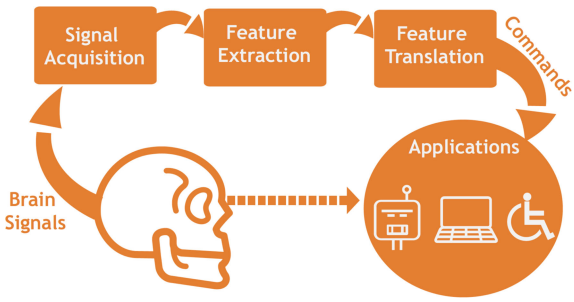

Fig. 1. BCI system design adopted from [7].

platforms' performance results are discussed. A bibliometric review of previous BCI software platforms is presented in Section III. Section IV contributes a discussion on web-based BCI, which is not included in earlier reviews. In Section V, we share current challenges and limitations. Suggestions for future BCI software platforms are provided in Section VI. Finally, Section VII concludes this article.

II. BCI Software Platforms

BCI system designs differ slightly across the literature. For example, Kothe and Makeig [28] suggest that a BCI system consists of five consecutive stages: signal acquisition, preprocessing or signal enhancement, feature extraction, classification, and control interface. However, Venthur [29] describes a system that consists of three stages: signal acquisition, signal processing, and feedback/stimulus presentation. Multiple other descriptions exist, but most are similar to the framework presented by [7]. This article classifies BCI systems into the following key phases: signal acquisition, feature extraction, feature translation, and commands/applications (see Fig. 1).

The initial signal acquisition phase usually outputs raw brain signals. Not much can be done with unprocessed brain signals. Due to the high dimensionality of brain signals and limited knowledge of how the brain works, translating brain signals into useful output can be challenging. BCI software that is flexible, easy to use, efficient, accurate, and robust is needed to address these challenges [30]. The following sections discuss the backgrounds and adoption of BCI software platforms. All citation information was collected via Thomas Reuters' Web of Science (WOS) Core Collection or Google Scholar when data were not available via WOS. The research area graph reflects the percentage of citations from each research area (see Fig. 3). Analysis of software downloads is also discussed when the data were available. All data reported in the following sections were retrieved on January 29, 2019.

A. BCI2000

1) Background: BCI2000 emerged in the early 2000s in response to a lack of general-purpose BCI tools [10]. The idea of creating a general-purpose tool was not a new concept when BCI2000 was introduced. Multiple papers discussing this issue were published around the same time. The main goals of BCI2000 were to provide researchers with a flexible BCI software platform and standard tools for BCI research. Prior to BCI2000, many experiments were done using highly customized systems that depended on specific BCI parameters. Unlike many of these earlier systems, BCI2000 provides a generic structure that utilizes a modular software design approach. As a result, various BCI protocol designs can be used with BCI2000 without having to make changes to the core software modules. BCI2000 was written in C++, which allows it to run efficiently on most systems. It is also compatible with MATLAB and Python. This compatibility is provided through online signal processing scripts written for these languages. It currently runs on Linux, Windows, and Mac OS. Additional details regarding BCI2000's system design are discussed in [11] and [27].

2) Adoption: According to WOS, the initial article describing BCI2000 [10] has been cited in more than 600 publications. As Fig. 2 illustrates, the adoption of BCI2000 steadily increased after it was released. This figure reflects 656 articles listed in WOS's Core Collection that cite [10]. Previous research citing BCI2000 contributed work to various research areas. Fig. 3 illustrates these areas according to WOS. This datum suggests that work citing BCI2000 is commonly focused in areas such as neuroscience, biomedical engineering, clinical neurology, multidisciplinary science, rehabilitation, psychology, neuroimaging, computer science artificial intelligence (AI), electrical engineering, and mathematical computational biology.

B. BioSig

1) Background: BioSig was released during the early 2000s. It was initially released as a tool for offline signal analysis. The first releases were based on MATLAB and Octave. In 2004, a package named real-time brain–computer interface (rtsBCI) was released, which enabled real-time data acquisition, storage, signal processing, and visualization based on MATLAB/Simulink. Much of BioSig's functionality is open source, except for a few optional MATLAB components. Although BioSig was not released until 2003, work leading up to BioSig can be traced back to 2001[9]. Developers of BioSig state its main goals are to address the lack of standardized approaches to BCI development and collaboration.

Multiple BioSig packages are available depending on user's preferences and needs. The most popular option, based on download numbers from SourceForge, is the BioSig for Octave and MATLAB (biosig4octmat) package. This package provides a toolbox for Octave and MATLAB with data management (import, export, etc.), feature extraction, classification, visualization, and scoring functionalities. Octave is a completely free and open-source high-level interactive language that is compatible with MATLAB. Additional details regarding BioSig's system design are discussed in [31].

2) Adoption: Evaluations of previous BCI software platforms mainly consider the number of citations as an indicator of adoption. Download information for BioSig is also publicly available. This information can be retrieved via BioSig's SourceForge page. According to SourceForge's site, BioSig packages have been downloaded more than 100 000 times since 2003. The 2008 BioSig article [32] is cited in more than 100 publications according to Google Scholar. A later BioSig article [31] is also cited in more than 100 publications according to Google

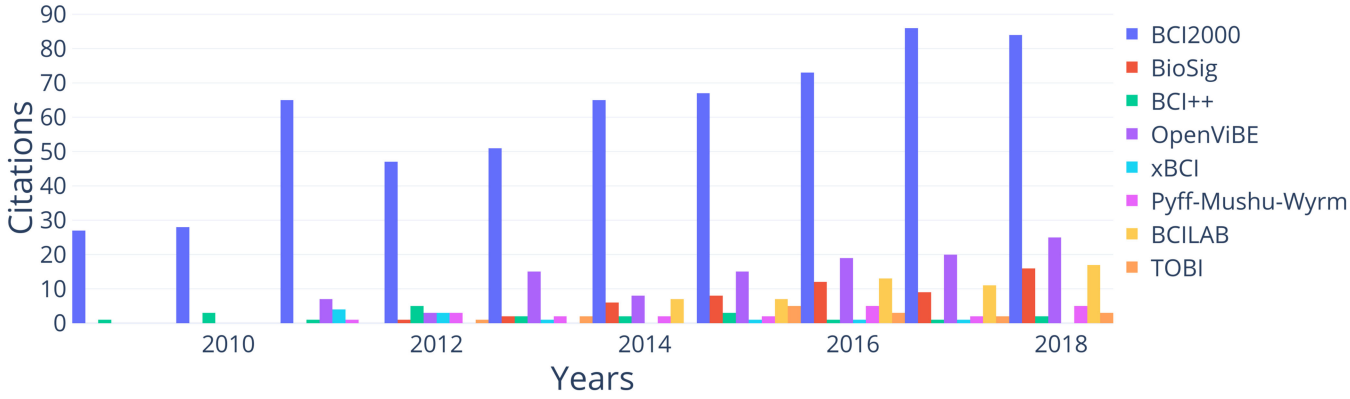

Fig. 2. Articles citing BCI software platforms (WOS) since 2009.

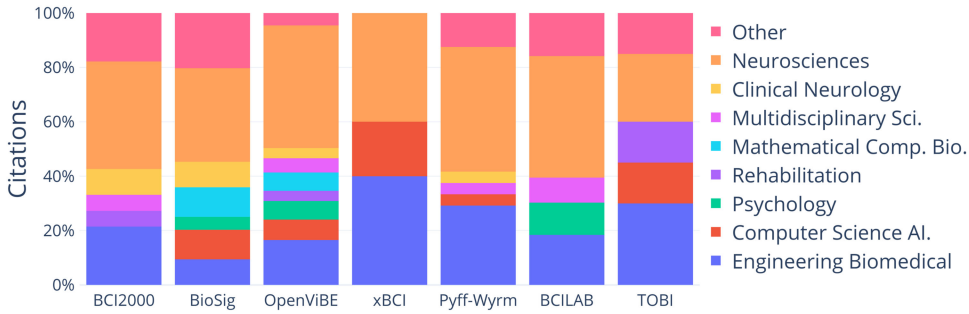

Fig. 3. Research areas of work citing BCI software platforms (WOS).

Scholar. Fig. 2 shows the 53 articles listed in WOS for the later articles since 2009. Fig. 3 reports data regarding the research areas of articles citing [32]. The research areas of articles citing BioSig include neuroscience, mathematical computational biology, computer science AI, biomedical engineering, clinical neurology, medical informatics, psychology, electrical engineering, chemistry, and more.

C. BCI++

1) Background: BCI++ was developed in response to the lack of emphasis on human–computer interaction in the BCI community [33]. BCI++ aimed to cater to both end-users in their everyday activities and researchers interested in developing new algorithms, protocols, and software modules. Along with accommodating end-users and researchers, BCI++ also aimed to simplify the process of interfacing with external devices. This platform featured a graphical user interface (GUI) tool called AEnima. This tool attempted to simplify the creation of new operating protocols for BCI applications and research experiments. The system uses a simple modular design consisting of two modules: AEnima and HIM. Each module handles separate functions and communicates with the other module via a Transmission Control Protocol/Internet Protocol (TCP/IP) connection. Additional details regarding BCI++'s system design are discussed in [11] and [27].

2) Adoption: Adoption of BCI++ has been limited in the research domain despite its focus on easing the development of BCI systems. The publications that introduce BCI++ [33],[34] have in total been referenced in 22 publications according to Google Scholar. Fig. 2 reflects work citing BCI++ each year since 2009. Due to the lack of data on the BCI++ platform via WOS, we do not provide an analysis of the research areas citing BCI++. Although the adoption of BCI++ has been limited, it is often still referenced as a relevant BCI software platform within the BCI community.

D. Open Virtual Brain Environment (OpenViBE)

1) Background: The OpenViBE was released in 2009, with the development of this software beginning in 2006. The goal of OpenViBE was to assist virtual reality (VR) developers, clinicians, and BCI researchers with designing, testing, and using BCI applications [35]. Similar to BCI++ (released around the same time) OpenViBE was designed to ease the development of BCI applications. Unlike many other BCI platforms, OpenViBE allows users to create complete scenarios using a graphical language. As a result, prior programming knowledge is not required to create basic BCI applications. This was a fairly novel concept for BCI development during that period. The few notable platforms released prior to OpenViBE (BCI++, BCI2000, and BioSig) all required some form of text-based programming to develop complete BCI applications. This platform also provides a complete software development kit (SDK) for programmers desiring to add new custom functionalities [36]. OpenViBE was built as a modular system using free portable C++ libraries. Additional details regarding OpenViBE's system design are presented in [11] and [27].

2) Adoption: Multiple researchers have adopted OpenViBE. Since its release in 2009, it has been cited in 111 publications according to WOS Core Collection. Fig. 2 shows the number of articles citing OpenViBE [35] each year since 2011. According to WOS, research areas of articles referencing OpenViBE include neuroscience, engineering biomedical, computer science AI, mathematical computational biology, psychology, rehabilitation, clinical neurology, and more.

E. xBCI

1) Background: xBCI emerged in 2009 as a generic BCI platform for online BCI analysis and experimentation [38]. This platform aims to provide users with a fast and easy-to-use BCI system. xBCI was designed for systems requiring acquisition and processing of at least 16 channel inputs. It features multithreaded parallel processing, high-speed data processing, GUI-based system realization, and an extendable module-based system design. Additional details regarding xBCI's system design are presented in [11] and [27].

2) Adoption: Only a few research articles have referenced xBCI. Although the limited amount of research provides a single perspective of adoption, this is just one of a few ways to analyze the platform with other existing systems. Currently, xBCI has been cited in four publications according to WOS and 12 according to Google Scholar. Fig. 2 shows the number of articles citing xBCI [38] each year since 2011 according to Google Scholar. Similar to BioSig, xBCI is also hosted via SourceForge. Since xBCI's release in 2009, it has received more than 1700 downloads. Articles citing the xBCI publication were from neurosciences, engineering biomedical, and computer science AI research areas.

F. Pyff, Mushu, and Wyrm (PMW)

1) Background: PMW were built in Python [29]. Together these components can be used to create a Python-based BCI. This platform appeared when interest in Python began to grow within the neuroscience community. Many BCI platforms prior to this platform required users to create custom BCI applications using C++ programming language, which could be a tedious process for nonprogrammers. Python is often considered easier to learn than C++ and more preferable to users new to programming. These components aim to leverage the advantages of Python to ease the process of developing BCI applications.

This BCI system features three separate parts released between 2008 and 2015. The Pythonic feedback framework (Pyff) component was released first [40], [41]. Pyff assists users with developing feedback applications and stimulus presentations. The main goal of this system was to provide a convenient process for programming BCI experimental paradigms that leverage visual or auditory presentation. Additional details regarding Pyff are presented in [11] and [27]. Afterward, Venthur and Blankertz presented Mushu, a free and open-source BCI signal acquisition software written in Python [42]. Mushu's main goal is to provide a unified interface to EEG data from multiple signal acquisition devices. Wyrm, an open-source BCI toolbox, was the third component presented [39]. This component handles the signals processing responsibilities of the system.

2) Adoption: Adoption of each component of this system has varied, which makes it slightly challenging to directly measure the influence of this system as a whole. However, combining bibliometric information for all articles that mention any component may offer a general idea of the platform's influence. Fig. 2 shows the sum of articles citing PMW. Data for Pyff and Wyrm were taken from WOS. Mushu citation data were taken from Google Scholar due to missing data in WOS. The common research areas of articles citing Pyff and Wyrm include neuroscience and biomedical engineering. The Python design strategy was also recently used to design deep learning BCI software [43].

G. BCILAB

1) Background: BCILAB was released in 2010 as an extension to the Swartz Center for Computational Neuroscience software [28]. One of the main focuses of BCILAB is assisting researchers with creating new BCI approaches and implementations. In contrast to other BCI platforms, BCILAB focuses less on out-of-the-box support for acquisition hardware and stimulus presentation. Overall, BCILAB aims to narrow the gap between cutting-edge method-oriented and application-oriented researchers. To assist with achieving this as well as application development, both a GUI and a scripting interface are provided with BCILAB. It also contains plug-in frameworks for method developers wanting to create rapid method prototypes. Details regarding BCILAB's system design are presented in [11] and [27].

2) Adoption: In 2013, BCILAB was one of the most comprehensive open-source machine learning frameworks hosted online[1] (a web-based machine learning open-source database) [28]. According to Google Scholar, BCILAB has been referenced in 123 articles. On WOS, it has been referenced in 54 articles. As shown in Fig. 3, the common research areas of articles citing BCILAB include neuroscience, biomedical engineering, rehabilitation, psychology, and more.

H. TOBI

1) Background: The Tools for Brain–Computer Interaction (TOBI) Common Implementation Platform (CIP) is a combination of interfaces that can be connected to create or extend a BCI system [85], [86]. The main goal of this system was to target people interested in merging BCI with other types of technology. It was introduced in 2010 as a system that facilitates hybrid BCI (hBCI). hBCI combines various input modalities with BCI. By utilizing multiple types of input channels, hBCI systems can switch between modalities based on a channel's reliability. Additional details regarding TOBI's system design are presented in [11] and [27].

2) Adoption: TOBI's interfaces help facilitate BCI research that utilizes multiple systems or hardware devices. TOBI CIP, similar to BioSig and xBCI, is hosted on SourceForge. Based on data provided via this site, TOBI has accumulated more

[1][Online]. Available: http://www.mloss.org

TABLE I
BCI SOFTWARE PLATFORMS PERFORMANCE MEASUREMENTS DISCUSSED IN THE PREVIOUS LITERATURE

Platform	Year	Task	Signal Processing	Sampling Rate [Hz]	Channels	Processing Latency [ms]	Jitter [ms]	Processor load[%]	EEG Hardware	Computer	CPU Speed
BCI2000 [37]	2010	MI	CAR, AR	512	16	8.02	0.46	NR[1]	g.USBamp	Mac Pro[2]	Quad-Core Intel Xeon 2.8 GHz, 8 gb RAM
xBCI [38]	2010	MI	BP	1000	16	1.07	2.88	6	Nihon Kohden amplifier	PC Windows XP	Intel Celeron 1.6 GHz, 512 MB
OpenVibe [35]	2010	MI	BP, LDA	512	10	NR	NR	7.17	NR[3]	GNU/Linux Fedora Core 6	Intel Xeon 3.80 Ghz, 4 GB RAM
BCILAB [28]	2013	MI	CSP	250	32	5.0–8.0	NR	NR	NR	NR	Dual-core Intel i3
Wyrm [39]	2015	ERP	LDA	240	64	4.17	NR	NR	N/A[4]	NR	Quad-core Intel i7 2.8 Ghz

[1] Processor load reported in the previous article [10]. [2] Dual booted with Windows XP. [3] Supported hardware discussed in [35]. [4] Simulated data used (see [39]). Acronyms: Autoregressive spectral estimation (AR), Band power estimation (BP), Common average reference (CAR), Common spatial pattern (CSP), Motor imagery (MI), Event-related potentials (ERP), Linear discriminant analysis (LDA), Data not reported (NR).

than 1200 downloads since October 2011. According to Google Scholar, TOBI has been referenced in 33 articles (see Fig. 2). On WOS, it has been referenced in more than 16 articles. Many of these works were published in areas such as biomedical engineering, neuroscience, rehabilitation, computer science AI, and others.

I. Performance

Each of the BCI software platforms discussed previously are primarily designed for operating systems that lack hard real-time capabilities [87]. Consequently, it is important that BCI software platforms describe performance results such as those shown in Table I. Previous work suggests that BCI systems must acquire and process brain signals within a short time period (i.e., milliseconds) [10]. This time requirement is vital as the tight relationship between users' intentions and feedback may significantly influence users' ability to master a BCI system. Strategies for addressing real-time requirements vary across BCI software platforms. For example, BCI2000 is designed to minimize dependency on lengthy operating system functions [10]. BCILAB leverages widespread use of vectorized code in an attempt to optimize real-time operations [28]. OpenViBE attempts to address issues with latency and jitter by enabling users to manually configuring drift correction via its acquisition server interface [88]. TOBI aimed to achieve optimal performance using thoroughly tested libraries from the boost library [85]. BioSig leveraged rtsBCI a MATLAB module based on Simulink and the Real-Time Workshop from Mathworks [89].

Information regarding BCI software real-time capabilities is often limited [12]. However, Table I offers a glimpse of BCI software platforms' soft real-time capabilities. It is important to note that the data presented in Table I are limited to information provided in previous articles. Table I data suggest that BCI software platforms processing latencies are usually below 10 ms. Furthermore, reports of system jitter and processor load are more infrequent in comparison to other system details. The adoption of standard procedures for systematic evaluations of BCI systems may assist with advancing future research on BCI software platforms.

J. Discussion

While there are similarities across current BCI software, specific platforms may be better equipped to handle certain scenarios. Consequently, BCI software platforms often feature various advantages and disadvantages. Furthermore, the prioritization of features may differ across various application scenarios and target users. For example, an experienced BCI researcher may highly prioritize the ability to leverage a broad set of classification algorithms (e.g., BCILAB, BioSig). Conversely, a nonexpert with limited experience may find value in using a visual environment or scripting language to develop a BCI prototype (e.g., OpenVibe, Wyrm). Clinicians working with clinical populations may highly prioritize thoroughly tested BCI systems (i.e., BCI2000). The following sections briefly discuss select advantages and potential limitations of each BCI software platform reviewed earlier.

1) BCI2000: As mentioned earlier, one of the BCI2000's key features is flexibility. BCI2000 mainly consists of a highly modular design. It was also written in C++, which allows it to run efficiently on most systems. BCI2000 is compatible with MATLAB and Python through signal processing scripts. Currently, BCI2000 runs on Linux, Windows, and Mac OS. To sustain its modular design, a TCP/IP-based protocol is used for communication between BCI2000's core modules (e.g., source, signal processing, and application) [10]. This approach allows users to make changes in specific modules without having to change the entire system. BCI2000 offers support for both online and offline processing and provides support for common BCI protocols such as P300, sensory-motor rhythm (SMR), and steady-state visual evoked potential (SSVEP). This platform also has an active and engaged open-source community as shown in Table II.

BCI2000 offers many advantages. However, it may present barriers in specific scenarios. While there is an option to write Python and MATLAB scripts, these options seem to target experienced developers. This approach could present barriers to entry for novice programmers. Additionally, BCI2000 offers limited support for open-source numerical analysis software alternatives to MATLAB (i.e., Octave). In comparison to other

TABLE II
BCI SOFTWARE PLATFORMS FEATURES AND OPEN-SOURCE ACTIVITY

	Online	Offline	Open	Visual	Languages	Protocols			Open Source	
						P300	SMR	SSVEP	Contributors	Commits
BCI2000	✓	✓	✓	✗	C++, MATLAB, Python	[45], [46], [47]	[48], [49], [50]	[51], [52], [53]	46	6017
BioSig	✓	✓	✓	✗	C/C++, MATLAB, Python, Java, R		[54], [55] [56]	[57], [58]	2	254
BCI++	✓	✗	✗	✗	C++		[33]	[59], [33]		
OpenViBE	✓	✓	✓	✓	Lua, Python	[60], [61], [62]	[63], [64], [65]	[66], [67], [68]	44	6678
xBCI	✓	✗	✓	✓	C/C++		[38]	[38]	1	1
BCILAB	✓	✓	✓	✓	MATLAB	[69]	[70], [71], [72]	[73][1]	4	322
TOBI[2]	✓	✗	✓	✗	C++	[74], [75]	[76], [77], [78]		12	1480
PMW	✓	✓	✓	✗	Python	[79], [80]	[39]	[81]	5	545

[1]Used both OpenViBE and BCILAB. [2]Open-source data combined from multiple TOBI tools [81]–[84].

platforms discussed in this article, BCI2000 offers support for a somewhat limited set of file formats (i.e., BCI2000, European Data Format (EDF) [90], General Data Format (GDF) [90], and MATLAB). However, these formats are popular in BCI research.

2) BioSig: One key goal of BioSig is to present a standardized approach to BCI application development. This aim was inspired by the lack of publicly available biomedical signal processing algorithms in the 1990s [92]. Currently, BioSig provides an open-source biomedical signal processing library compatible with both MATLAB and Octave. It also provides support for more than 40 different data formats. These formats include commonly used file types such as BCI2000 native files, GDF, EDF, Extensible Biosignal, BioSemi Data Format, Standard Communication Protocol for Electrocardiography, File Exchange Format, HL7 formatted files (Extensible Markup Language based biosignal files), and MATLAB files. In addition to the BioSig for Octave and MATLAB package, other BioSig packages include BioSig for C/C++ (biosig4c++) and the rtsBCI system. The biosig4c++ package provides reading and writing capabilities for various types of biosignal data formats. The rtsBCI package is used for conducting real-time BCI experiments. This package is implemented in MATLAB and Simulink. Currently, the BioSig project includes support for multiple languages such as C/C++, MATLAB, Octave, Python, Java, and R.

Similar to BCI2000, BioSig targets experienced programmers. While components such as BioSig's SigViewer may assist with visualizing and scoring EEG data, developing more complex applications with BioSig may require additional programming experience. Based on the data retrieved from BioSig's Sourceforge repository shown in Table II, the project has had limited open-source activity relative to other BCI software platforms discussed in this article. There also seems to be a lack of previous P300 work that references BioSig. This suggests that other BCI software platforms may have been preferred for P300 applications. It must be noted that some authors simply may not have mentioned BioSig explicitly in their work.

3) BCI++: BCI++ aims to address the needs of both researchers and end-users of BCI applications. This approach resulted in BCI system components that focused on the design and development of end-user applications. This approach differs from previous work that primarily strives to enhance signal acquisition and algorithm development aspects of BCI systems. For example, BCI++'s AEnima component features modules for two-dimensional/three-dimensional graphics, audio engine, and event management. Each of these modules features high-level functions to assist developers in creating virtual environments. The authors of this article also discussed early examples of using the event manager module to communicate BCI output to external systems such as a home automation system.

BCI++ presented an early attempt to support end-users with BCI systems. However, nonexpert developers unfamiliar with C, C++, and networking concepts may encounter challenges while creating new BCI applications. There is also limited information available regarding open-source activity associated with BCI++. Additionally, there is limited previous P300 and SMR research that explicitly references BCI++.

4) Open Virtual Brain Environment: Similar to BCI++, OpenViBE aims to ease the development of BCI applications. To assist with this goal, OpenViBE features a graphical language that may assist nonexpert programmers with constructing BCI pipelines. OpenViBE also provides a complete SDK for programmers desiring to add new custom functionalities. In addition to the graphical interface, users may also write OpenViBE scripts in Lua or Python. Embedded VR tools for users interested in using VR displays for feedback are also provided. As shown in Table II, the OpenViBE open-source community has been active since its initial release.

Users with limited programming experience may find value in using a visual environment to develop a BCI prototype. However, experienced BCI researchers may highly prioritize the ability to create and investigate a broad set of signal processing and classification algorithms. In the latter case, developers may prefer a traditional text-based approach. This approach may be especially true for BCI researchers primarily seeking to validate processing and classification algorithms offline. In general, text-based BCI software platforms may be more advantageous for these types of use-cases.

5) xBCI: Similar to many other BCI software platforms previously discussed, xBCI is built with C/C++. It also features a modular design made possible by the transmission of information via TCP/IP or user datagram protocol (UDP). This results in a flexible system consisting of multiple independent modules. Changes to these modules can be made separately without having to rebuild the entire system. xBCI also features ready-to-use BCI components that can be arranged in a GUI-based diagram editor. These components handle BCI functions such as signal acquisition, storage, mathematical operations, signal processing,

network communication, data visualization, experiment control, and real-time feedback presentation.

Similar to OpenViBE, xBCI aims to leverage a graphical approach to assist developers in creating new BCI applications. This strategy may not effectively address experienced BCI developers' goals related to flexibility. xBCI supports the development of custom components via C++ or QtScripts. However, to our knowledge, it does not provide support for other popular scripting languages such as Python. There is limited information regarding open-source activity related to this project. As shown in Table II, the use of xBCI in previous BCI research seems limited. It should be noted that previous work may have simply failed to reference xBCI explicitly.

6) BCILAB: BCILAB provides an open toolkit for BCI methods research and evaluation. Currently, BCILAB is compatible with Windows, Linux, and Mac OS. It is primarily designed to assist researchers with investigating cutting-edge methods for developing BCI applications. BCILAB features both a GUI and a scripting interface via MATLAB. This approach may help researchers studying various areas of science and engineering such as signal processing, machine learning, experimental neurosciences, and machine learning. BCILAB features more than 100 preimplemented algorithms with an emphasis on the research, development, and testing of advanced BCI methods. This project has had a moderate level of open-source activity.

Contrary to other BCI software platforms discussed previously, BCILAB focuses less on out-of-the-box support for acquisition hardware and stimulus presentation. Researchers or practitioners seeking native support for signal acquisition hardware or user feedback components may prefer alternative BCI platforms. Unlike many other BCI platforms, the design of BCILAB is not focused on modularity. This approach may present challenges for developers seeking to build custom distributed real-time BCI applications. However, BCILAB does provide support for Lab Streaming Layer [93], which can be used to integrate BCILAB with a broader range of external signal acquisition and stimulus-presentation components.

7) TOBI: TOBI is a cross-platform library developed in C++. One of the main goals of the TOBI project is to combine traditional input modalities with BCI. In result, the design of TOBI is focused on modularity and flexibility. TOBI consists of four components that separate concerns related to signal transmission, feature extraction, and classification. This approach may assist developers in needing to integrate BCI into existing systems. TOBI also aims to support the concept of fusion and shared control between BCI and various other types of input modalities (e.g., mouse, keyboard).

Similar to many other BCI platforms, TOBI may be an ideal option for experienced developers wanting to build distributed BCI applications that feature multiple types of input modalities. However, it may present challenges to novice programmers. Currently, there is no native support for scripting or a GUI environment to assist nonexpert programmers with getting started with TOBI. Furthermore, researchers more interested in the research and design of useful signal processing or classification algorithms may not require components that assist with implementing fusion and shared control features.

8) Pyff, Mushu, and Wyrm: Researchers and practitioners that are more familiar with Python than C/C++ may prefer building simple prototypes with PMW. The modular design of the three separate components enables users to select the functionality required for their project. For example, a researcher may choose only to use Pyff [40], [41] if only the stimulus presentation is desired. Mushu [42] may be leveraged individually in cases where a developer is only interested in signal acquisition. Additionally, Wyrm [39] could be utilized when signal processing and feature classification functionalities are needed. The Pyff component also features a GUI environment and feedback controller that allows users to load and control feedback to the target user.

Researchers and developers interested in integrating BCI into existing Python projects may find this platform particularly useful. In contrast, developers seeking to leverage performance advantages provided by C and C++ may favor one of the platforms discussed previously. This project's highly modularized design offers some benefits. However, researchers seeking a seamless out-of-the-box experience may prefer an alternative BCI software platform. BCI software platforms that aim to assist nonexpert programmers commonly offer both GUI-based and scripting features. While Pyff provides a GUI for stimulus-presentation development, Mushu and Wyrm do not provide graphical support for signal processing and feature classification tasks.

9) Summary: The systems discussed previously are not always used independently. For example, researchers have previously combined BCILAB and OpenViBE to investigate an SSVEP BCI application [72]. Similar studies have also leverage components of BCI2000 and BioSig together. This pattern suggests that a unified open-source framework for BCI application development may assist with advancing BCI research in the future. Ideally, this approach would leverage existing BCI efforts in areas such as signal processing, scripting, and visual programming. Additionally, this approach may help reduce the fragmentation of efforts across multiple BCI software platforms. Table II provides a summary of features currently offered by the BCI software platforms discussed in this article. Data related to relevant open-source projects' activity are also provided. Specifically, the number of contributors and commits are reported. These totals suggest that projects such as BCI2000 and OpenViBE have engaged an active community. Unsurprisingly, there also seems to be a relationship between the number of contributors, commits, and citations (see Fig. 2). Additional research may provide further insights on how open-source BCI projects influence BCI development.

III. Bibliometric Review

Although a number of platforms report data (page hits, downloads, etc.) specific to their tool, there is limited literature available that provides information about BCI researchers' background and topics of interest throughout the progression of BCI software platforms. To assist with addressing this knowledge gap, bibliometric data on publications referencing the BCI platforms reviewed in this article were retrieved via WOS and

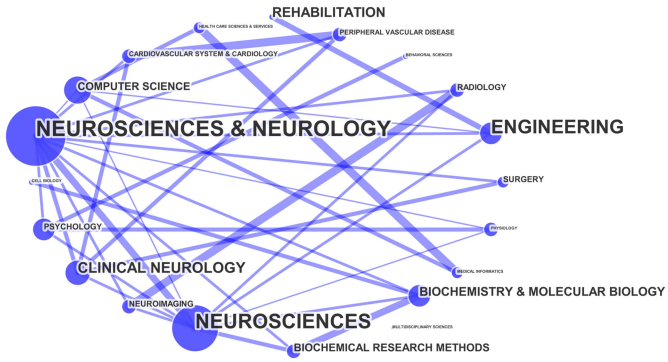

Fig. 4. Network (based on cocitation analysis) between areas in most cited work during Phase I (2000–2008) of BCI platforms.

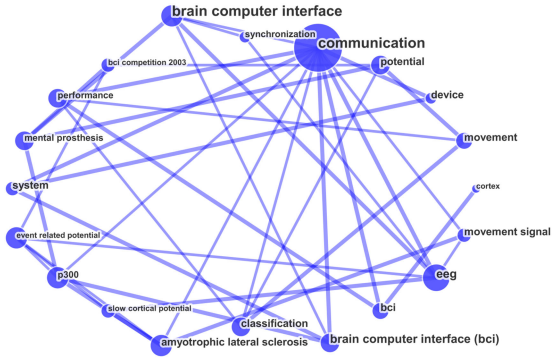

Fig. 5. Network (based on cocitation analysis) between terms in most cited work during Phase I (2000–2008) of BCI platforms.

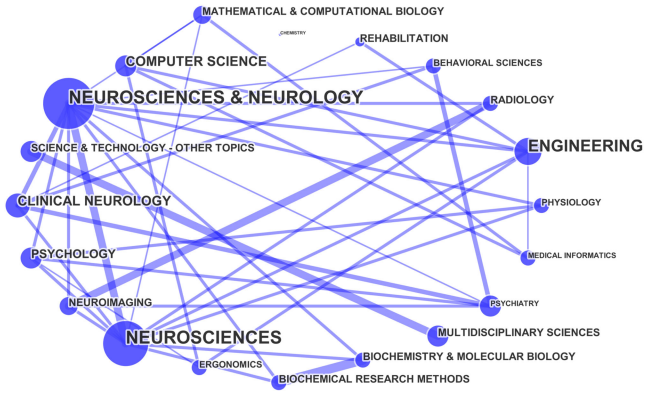

Fig. 6. Network (based on cocitation analysis) between areas in most cited work during Phase II (2008–2018) of BCI platforms.

analyzed using Citespace [94]. A total of 756 papers, retrieved via WOS, were used to provide a bibliometric review. Figs. 4–7 present the visualizations of nodes corresponding to cocitation networks. These cocitation analysis figures illustrate networks corresponding to the most cited BCI literature. The size of each node provides a general idea of an item's rank according to CiteSpace's calculation. This rank takes into account patterns such as the number of times works are cited. Multiple occurrences of an item in the same paper were only counted once. The sizes of the edges connecting two nodes indicate how frequently

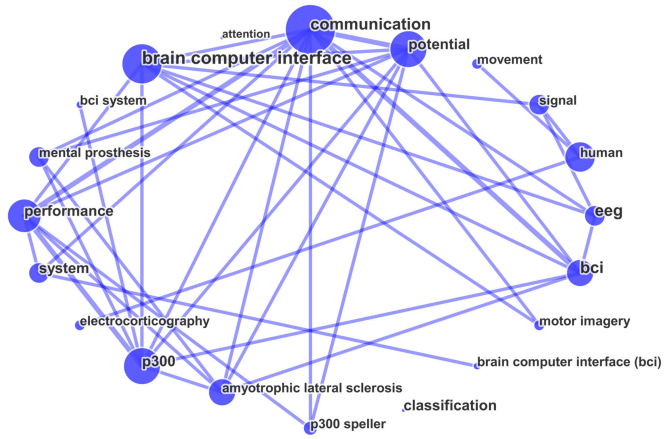

Fig. 7. Network (based on cocitation analysis) between terms in most cited work during Phase II (2008–2018) of BCI platforms.

this pattern occurred. This approach was also used to analyze patterns across key terms and research categories. Availability of modern standardized BCI platforms presented in this article can be analyzed in two phases: Phase I (2000–2008) and Phase 2 (2008–2018).

A. *Phase I*

During Phase I, BCI2000, BF++ (not thoroughly covered in this article due to its low citation count), and BioSig provided the initial steps toward standardized BCI tools. Multiple BCI platforms became available during the earlier period of this phase (2000–2003). During the later half of this period, no significant alternative platform was introduced. Fig. 5 provides a glimpse of key terms that appear in the literature involving the BCI platforms reviewed in this article. This information could potentially provide a broad idea of the goals and needs of BCI researchers during Phase I. For example, Fig. 5 illustrates a strong emphasis on communication present in literature during Phase I. This may reflect the medical focus of researchers during this period. During this phase, a large majority of researchers investigating BCI had goals that aligned with providing communication and control/movement abilities to the motor-impaired population. amyotrophic lateral sclerosis (ALS), a severe form of neuromuscular disorder, was also an issue targeted by many BCI researchers during this time. As shown in Fig. 5, to address these issues, researchers utilized techniques and brain characteristics such as p300, event-related potentials, and slow cortical potentials. The nature of EEG synchronization (neuronal activity occurring together during certain tasks) was also used to gain knowledge on how to address BCI research issues. A sample dataset provided during a BCI competition in 2003 [95] was also among the most highly referenced topics during this phase. This dataset provided a way for researchers to test and compare BCI methods during this phase. Concerns such as performance and classification were also popular during this phase.

As new BCI concepts and methods emerged, so did the idea of adopting multidisciplinary approaches to problems. As shown in Fig. 4, work referencing BCI platforms involved researchers from various disciplines. Among the most influential areas

were neuroscience, engineering, clinical neurology, rehabilitation, biochemistry and molecular biology, computer science, psychology, and physiology. This multidisciplinary approach was vital for progress in BCI research during Phase I. The type of users and researchers that contributed to these areas include neuroscientists, biomedical engineers, clinicians, rehabilitation specialists, electrical engineers, and signal processing engineers. Foundational knowledge offered from neuroscientists, clinicians, and rehabilitation specialists has always been critical to BCI. New collaborations between these researchers and computer scientists at the turn of the 20th century may have had an influence on the field of BCI. Multiple concepts may have become common in BCI due to this collaboration. One of the most significant concepts was modularity. As stated by [6], the field needed a new functional model for BCI system design. This need was addressed through modularization, a software engineering principle introduced in the field of computer science in the early 1970s [96]. This approach was taken with most general-purpose BCI platforms released over the past decade, as discussed in the BCI software platform section. Advances made during the dot-com boom may have also influenced BCI software platforms. The TCP and UDP web communication protocols played a critical role in the software design of many BCI platforms released during Phase I. These communication protocols often served as information passageways between various components of BCI software platforms. Hence, novel BCI applications were possible due to the flexibility and modularity offered by this new approach. Along with these advances, BCI researchers and developers also began to embrace the concept of open-source software (see Table II). This approach resulted in the emergence of BCI developer communities that gradually improved BCI platform efficiency, stability, and extendibility. An additional trend that emerged across Phase I BCI platforms was a dependence on MATLAB as it provided the foundation for most of the BCI platforms released during Phase I. MATLAB was beneficial for tasks such as matrix calculations of raw EEG data, visualizations, and custom scripting. The progress made by the general-purpose BCI platforms during Phase I influenced the design of platforms released later during Phase II.

B. Phase II

During Phase II, BCI++, OpenViBE, Pyff, xBCI, BCILAB, and TOBI were released. Most of the platforms released during this phase appeared between 2009 and 2011. Although no complete platform was released between 2011 and 2018, Venthur and Blankertz released Mushu [42] and Wyrm [39]. As previously mentioned, these components can be merged with Pyff to provide a full Python-based BCI system.

Most of the BCI researchers' goals and needs during Phase II were similar to those of Phase I. For example, as illustrated by Fig. 7, terms such as communication, ALS, and performance remained popular in BCI platform literature during Phase II. However, new terms such as human and attention became popular. This is likely due to a turn toward more human-centered approaches in BCI research by multiple researchers [19], [21]. Fig. 4 also hints toward this shift to areas such as ergonomics and multidisciplinary sciences. Another area that became popular in BCI research was mathematical and computational biology. Work from this area was often referenced along with computer science and medical informatics, and neuroscience. This change indicates a shift toward more data-driven approaches to BCI research during Phase II.

To accommodate the new research interests, BCI platforms began to transform. Part of this transformation was driven by two key design concerns: ease of use and flexibility. As discussed earlier, Phase I consisted mostly of engineers, computer scientists, and research with backgrounds related to neuroscience. However, during Phase II users outside of these fields gained interest in leveraging the capabilities of BCI. Accordingly, a new class of BCI researchers with different backgrounds emerged. In contrast to BCI platforms released during Phase I, BCI platforms that required less programming skills began to appear. For example, OpenViBE features an interface that allows users to create complete BCI scenarios using a visual programming environment. The platform presented by Venthur *et al.* allowed users to create custom BCI solutions using Python, a programming language that is often considered easier to learn than other languages such as C++. xBCI also features a GUI-based interface that allows users to arrange its components. These are a few of the features that appeared due to the new emphasis on ease of use during Phase II. Flexibility also played a major role in Phase II BCI platforms. As researchers from diverse backgrounds started to get involved with BCI research, applied uses began to take on new forms. For example, applications such as controlling quadcopters [97], pinball machines [98], and painting [99] that emerged in Phase II could benefit from flexibility much more than many Phase I BCI applications. To develop these kinds of systems, it was vital that BCI platforms needed to offer researchers and developers the option to extend a platform's functionality. Many of the BCI platforms addressed this issue through customizable plug-in and modules. These plug-in and modules could be used to integrate various kinds of external interfaces. The approach of sharing information between the core BCI system and external devices using TCP and UDP also assisted with addressing the need of flexibility.

BCI platforms also began to gradually move away from a heavy dependence on MATLAB during Phase II. This was mostly due to a change in the desired functionalities of BCI platforms. While MATLAB greatly assists with tasks such as imagining and processing, other approaches may be more optimal for creating highly connected and flexible systems. Examples of these new approaches can be observed in TOBI's multimodule architecture design, PMW's multicomponent BCI, and OpenViBE's "box" concept.

IV. Web-Based BCI Software

A web-based BCI is a system that uses web technologies to facilitate interactions with neural activity. Reviews of traditional BCI software have been published previously. However, there are limited discussions on web-based BCIs. This section aims to address this gap by presenting various tools and implementations relevant to web-based BCI work. The following sections

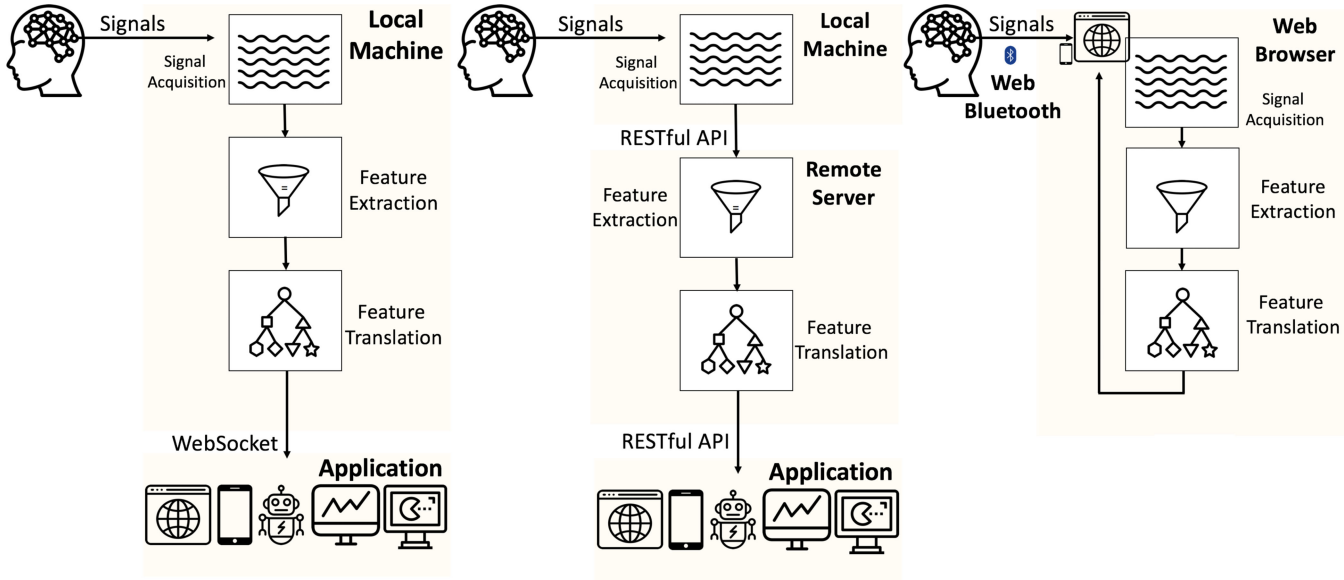

Fig. 8. Web BCI systems that perform feature extraction and translation procedures (a) locally, (b) on a remote server, or (c) in a web browser.

briefly discuss web-based BCIs in four stages: signal acquisition, feature extraction, feature translation, and applications.

A. Signal Acquisition

The signal acquisition stage of a web-based BCI system is often similar to non-web-based BCI systems. Raw signals are usually acquired and passed to the computer via universal serial bus (USB) or Bluetooth. One common approach is to acquire and process raw signals in a traditional desktop application before passing data to a web environment using WebSockets [106]–[109] [see Fig. 8(a)]. WebSockets support real-time bidirectional event-based communication between servers and web applications. Previous work has leveraged the Node.js [110] JavaScript runtime environment to implement web-based BCI systems. The approach shown in Fig. 8(a) may address the performance constraints of most web browsers. However, this occasionally requires users to install software (i.e., drivers) before collecting data from the BCI hardware. Furthermore, the required software may not be available for all operating systems. While this is likely not a problem for researchers familiar with BCI, it may present barriers for others.

Communication technologies such as Web Bluetooth [111] may present future opportunities for BCI developers to stream data directly to a web browser [see Fig. 8(c)]. Web Bluetooth allows web browsers to communicate with Bluetooth 4 devices using the generic attribute profile. Partial implementation of Web Bluetooth is currently available on Android, Chrome OS, Linux, Mac, and Windows 10 with Chrome 70 or greater. Currently, this is an experimental technology and not a W3C Standard. Previous work leveraging Web Bluetooth mostly investigated physical web applications [112]–[115]. However, this protocol may present new opportunities for future BCI software platforms. Developers have recently begun exploring Web-Bluetooth-based BCI applications with consumer-grade EEG technology [116]. Currently, there is a lack of research investigating the use of Web Bluetooth with clinical-grade EEG systems. Web Bluetooth may present advantages for end-users of BCI applications. However, tradeoffs between energy consumption, latency, and throughput may present challenges. Future controlled studies are needed to better understand the feasibility of Web Bluetooth as a means for data collection.

B. Preprocessing and Feature Extraction

Brain signals are often noisy and difficult to interpret. In most instances, this noise is a result of electrical activity arising from nonbrain artifacts. These artifacts can arise from events such as muscle movement (e.g., clenching jaw) or eye movements (e.g., eye blinks). Artifacts may contaminate brain signals and ultimately hinder the performance of a BCI system. To address this issue, brain signals go through multiple stages before being translated into a command. During the first stage, brain signals are preprocessed. Denoising methods are used during this step to reject nonbrainwave artifacts. Applying signal filtering techniques during this phase also reduces the signal-to-noise ratio.

Preprocessing and feature translation procedures can be integrated in web-based BCI systems using various approaches. Fig. 8 illustrates three methods that have been previously presented by developers and researchers. The most common approach involves handling preprocessing and feature extraction before transmitting data over a network to a web-based BCI application [see Fig. 8(a)]. This method allows a web-based BCI system to take advantage of numerous previously developed algorithms. This approach could assist researchers looking to extend existing BCI systems. Previous research on teleoperated brain–machine interface applications leveraged a similar design [117]–[120]. Recently, Milsap *et al.* presented the WSIOFilter, which allows raw and processed signals from BCI2000 to be transmitted to a web browser [107]. Handling preprocessing

TABLE III
WEB-BASED TOOLS SUPPORTING COMMON BCI FEATURE EXTRACTION AND CLASSIFICATION ALGORITHMS

Preprocessing and feature extraction	Classification
ICA [101], BP [101], [102]	LDA [101]
IIR [102], [103], FIR [103],	SVM [104]
PSD [101], [102], CSP [101]	KNN [104]
PCA [104]	Naive Bayes [104]
	NN [105], [106]

and feature extraction procedures with a traditional desktop application prior to communicating data over the network to a web-based BCI application could reduce the computational load of the client. However, users will likely need to install additional software before getting started with these BCI systems.

An alternative web-based BCI design involves communicating signals over WebSockets prior to applying preprocessing or feature extraction procedures [see Fig. 8(b)]. Signals are then transmitted to remote servers equipped with advanced feature extraction and translation algorithms. This approach presents opportunities for new physiological data processing industries. For example, Intheon's NeuroScale product leverages a RESTful application programming interface (API) to communicate EEG data to biosignal processing tools in the cloud [121]. After the data are analyzed, feedback can be sent back to the client's device. This approach may be especially interesting for applications running on devices with limited computational power. However, it is not clear whether performance issues related to latency will influence web-based BCI systems. Furthermore, storage of EEG data on servers may cause potential data privacy concerns. Additional work at the intersection of neuroscience, technology, and ethics may help address unforeseen negative impacts of web-based BCI solutions.

Recent developments in web technologies have also enabled browser-based feature extraction and translation [see Fig. 8(c)]. This approach enables local in-browser computations that may increase data privacy. Browser-based feature extraction and translation might present performance issues when compared to the previous two alternatives. However, the recent development of web assembly, which allows compiled code to run directly in the web browser at near native speeds, may soon close this performance gap [122]. A transition from server-based processing to client-side processing has also been observed in other domains. For example, in 2019 Google AI published an article discussing their ability to perform speech recognition on-device [123].

Table III presents additional browser-based technologies that could be used to extract and translate features from EEG. While these JavaScript tools present opportunities for browser-based BCIs, further research is needed to validate the feasibility of this approach. Although web-based signal processing and machine learning libraries provide important first steps towards web-based BCIs, they may not provide the ideal solution for the training of models. As the training process for machine learning models often requires significant computational resources, offloading data to a more powerful server still provides an advantage in this circumstance.

C. Feature Translation

After brain signals are preprocessed and the desired features are extracted, translation algorithms can be used to detect and classify various brain states. Fig. 8 illustrates how feature translation is integrated in most web-based BCI systems. Recently, JavaScript libraries relevant to feature translation such as ml.js [103], BCI.js [100], and MLweb [124] have been presented. Furthermore, neural network libraries such as ConvNetJS [105], ml.js [103], and TensorFlow.js [125] have also emerged. Future browser-based BCIs could be realized through preprocessing, feature extraction, and machine learning JavaScript libraries. This method may be ideal for activities such as teaching and interactive demonstrations in remote areas with poor internet connections. Furthermore, tablet and smartphone web-based BCI applications could be further investigated using this approach.

D. Command and Applications

BCI systems use cognitive commands to support various types of applications. BCI applications tend to fall into the following categories: device control, user-state monitoring, evaluation, gaming and entertainment, and cognitive improvement. Device control applications usually consist of users directly controlling devices such as wheelchairs [13]–[15] or virtual keyboards [17]. Going forward, web libraries could be used to develop additional web-based applications. For example, researchers recently presented a browser-driven SSVEP-based BCI web speller [126]. The JavaScript runtime environment Node.js and package repository NPM [127] could also assist developers with designing numerous future BCI applications. Previous works has investigated the use of Node.js with BCI applications including educational technology [128], state monitoring [129], and musical expression [130]. Furthermore, JavaScript graphic libraries may provide browser-based feedback for BCI systems. For example, researchers have previously used JavaScript libraries to design a SSVEP BCI [131] and plot streaming data [132]. Further BCI research involving tools such as WebGL [133] is necessary to better understand the potential of future web-based BCI applications.

V. CHALLENGES AND LIMITATIONS

Although BCI platforms have made much progress over the past years, challenges still exist that hinder end-users [134]. One challenge is the relatively high barriers to entry present in many of the BCI platforms. Vital resources ranging from documentation on BCI platforms options to user-friendly getting started documents are currently missing or hard to locate online. Many of the resources provided by BCI platforms target end-users with technical backgrounds (developers, neuroscientists, etc.). These documents could be very difficult to follow by someone without a strong technical background. Even though steps have been taken by platforms to address these issues, many users may still have issues due to a lack of emphasis on out of the box functionalities. One of the key issues causing the barrier to entry problem is the need for end-users to make multiple

modifications before the BCI platform is usable. An additional issue is integrating new acquisition hardware, an issue especially prominent with newer consumer-grade BCI devices. Currently, end-users are often expected to connect these devices by developing custom plug-in or modules. This usually requires users to have programming experience. One proposed solution to this issue is sharing plug-ins online via sites intended for the open-source community. Also, there is no way to predict when new signal acquisition plug-ins will be contributed. Hence, support for new devices can be sporadic and unreliable. Although this is a step in the right direction, it still serves as a deterrent to many inexperienced users.

Many BCI platforms are often optimized for certain types of tasks. Currently, there is a lack of BCI platforms that easily handle a large span of BCI applications. For example, some of the BCI platforms provide extensive support for stimulus applications. Other BCI platforms, on the other hand, offer many different processing options but depend on a complex infrastructure for applications involving external devices. As BCI software platforms progress it may be useful to design BCI software platforms that reflect experiences users have working with other tools such as video editing or word processing applications. Future web-based BCI software platforms may assist with a few of these challenges. However, much is still unknown regarding the feasibility of these systems.

VI. Moving Forward

As the landscape of BCI users shifted over the past decade so did the features offered by BCI platforms. In many cases, these shifts came in response to changing goals and needs of BCI users. It could also be argued that the types of applications developed by BCI researchers were greatly impacted by available features.

One interesting ideological shift related to changes in BCI platform features was the use of BCI beyond the medical domain [19]. Although BCI was initially used to enable people with disabilities, BCI is gradually starting to appear in nonmedical applications. If sustained this shift could continue to shape the design of BCI software platforms in the future. Much of the previous efforts to move towards practical BCI applications focused on improving the usability of signal acquisition hardware. Therefore, multiple consumer-grade BCI devices are now available. Although usable signal acquisition hardware devices are becoming more readily available [12], improvement is needed in BCI software platforms. Improving end-users' interactions with BCI software could positively influence the wide spread usage of BCI technology. Increasing commercial interest has primarily fueled previous progress of usable signal acquisition hardware. Going forward a similar push may inspire the development of more usable general-purpose BCI software platforms. Along with commercial interest, continued emphasis on open-source BCI software platforms could assist with moving this effort forward. As previously discussed, recent advancements involved contributors worldwide. Without the support of active open developer communities, BCI software platforms may become stagnant. Unlike many early BCI software platforms that mainly included features geared toward communication and control functionalities, BCI platform developers may begin to embrace additional hybrid approaches. This shift involves the integration of BCI with other interfaces such as joysticks, mice, buttons, or eye trackers. To address this shift platforms can begin to provide features that make it easier for developers to create applications that seamlessly combine BCI and other types of input modalities. Further investigations of web-based BCI systems may also play a key role in addressing challenges going forward.

VII. Summary and Conclusion

Many BCI software platforms have emerged since the beginning of the 21st century. This article presented a bibliometric review and discussed the adoption of major BCI software platforms. An analysis of topics and research areas involved with BCI research was also presented in this article. This article also discussed emerging technologies relevant to web-based BCI systems. In general, it has been found that many previously published works have utilized software platforms such as BCI2000 and OpenViBE. Furthermore, BCI researched that leveraged software platforms frequently appear in areas related to neuroscience, biomedical engineering, and computer science according to the World of Science. Continual progress of BCI technology requires further investigation of many issues concerning current BCI systems. Some of the most prominent issues include software flexibility, extendibility, usability, and performance. Further research regarding these aspects of BCI software platforms could lead to interesting BCI applications in the future.

References

[1] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain–computer interfaces for communication and control," *Clin. Neurophysiol.*, vol. 113, no. 6, pp. 767–791, 2002.

[2] R. Caton, "Electrical currents of the brain," *J. Nervous Mental Disease*, vol. 2, no. 4, pp. 610–611, 1875.

[3] H. Berger, "The electroencphalogram of man," *Eur. Arch. 1027 Psychiatry Clin. Neurosci.*, vol. 87, no. 1, pp. 527–570, 1929.

[4] J. J. Vidal, "Toward direct brain-computer communication," *Annu. Rev. Biophys. Bioeng.*, vol. 2, no. 1, pp. 157–180, 1973.

[5] J. J. Vidal, "Real-time detection of brain events in EEG," *Proc. IEEE*, vol. 65, no. 5, pp. 633–641, May 1977.

[6] S. G. Mason and G. E. Birch, "A general framework for brain-computer interface design," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 1, pp. 70–85, Mar. 2003.

[7] D. J. McFarland and J. R. Wolpaw, "Brain-computer interfaces for communication and control," *Commun. ACM*, vol. 54, no. 5, pp. 60–66, 2011.

[8] A. Delorme and S. Makeig, "EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *J. Neurosci. Methods*, vol. 134, no. 1, pp. 9–21, 2004.

[9] C. Guger, A. Schlogl, C. Neuper, D. Walterspacher, T. Strein, and G. Pfurtscheller, "Rapid prototyping of an EEG-based brain-computer interface (BCI)," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 9, no. 1, pp. 49–58, Mar. 2001.

[10] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000: A general-purpose brain-computer interface (BCI) system," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1034–1043, Jun. 2004.

[11] C. Brunner *et al.*, "BCI software platforms" in *Towards Practical Brain-Computer Interfaces*. Berlin, Germany: Springer, 2012, pp. 303–331.

[12] P. Brunner, L. Bianchi, C. Guger, F. Cincotti, and G. Schalk, "Current trends in hardware and software for brain–computer interfaces (BCIs)," *J. Neural Eng.*, vol. 8, no. 2, 2011, Art. no. 025001.

[13] T. Carlson and J. del R. Millán, "Brain-controlled wheelchairs: A robotic architecture," *IEEE Robot. Autom. Mag.*, vol. 20, no. 1, pp. 65–73, Mar. 2013.

[14] I. Iturrate, J. M. Antelis, A. Kubler, and J. Minguez, "A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation," *IEEE Trans. Robot.*, vol. 25, no. 3, pp. 614–627, Jun. 2009.

[15] F. Galn *et al.*, "A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots," *Clin. Neurophysiol.*, vol. 119, no. 9, pp. 2159–2169, 2008.

[16] G. R. Müller-Putz, R. Scherer, G. Pfurtscheller, and R. Rupp, "EEG-based neuroprosthesis control: A step towards clinical practice," *Neurosci. Lett.*, vol. 382, no. 1, pp. 169–174, 2005.

[17] N. Birbaumer *et al.*, "A spelling device for the paralysed," *Nature*, vol. 398, no. 6725, pp. 297–298, 1999.

[18] E. Buch *et al.*, "Think to move: A neuromagnetic brain-computer interface (BCI) system for chronic stroke," *Stroke*, vol. 39, no. 3, pp. 910–917, Mar. 2008.

[19] J. van Erp, F. Lotte, and M. Tangermann, "Brain-computer interfaces: Beyond medical applications," *Computer*, vol. 45, no. 4, pp. 26–34, 2012.

[20] C.-T. Lin, C.-J. Chang, B.-S. Lin, S.-H. Hung, C.-F. Chao, and I.-J. Wang, "A real-time wireless brain–computer interface system for drowsiness detection," *IEEE Trans. Biomed. Circuits Syst.*, vol. 4, no. 4, pp. 214–222, Aug. 2010.

[21] E. Cutrell and D. Tan, "BCI for passive input in HCI," in *Proc. Conf. Human Factors Comput. Syst.*, 2008, vol. 8, pp. 1–3.

[22] D. Grimes, D. S. Tan, S. E. Hudson, P. Shenoy, and R. P. Rao, "Feasibility and pragmatics of classifying working memory load with an electroencephalograph," in *Proc. ACM SIGCHI Conf. Human Factors Comput. Syst.*, 2008, pp. 835–844.

[23] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun, "Emotion assessment from physiological signals for adaptation of game difficulty," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 6, pp. 1052–1063, Nov. 2011.

[24] D. Szafir and B. Mutlu, "Pay attention!: Designing adaptive agents that monitor and improve user engagement," in *Proc. ACM SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 11–20.

[25] H. Unterrainer, M.-L. Chen, and J. Gruzelier, "EEG-neurofeedback and psychodynamic psychotherapy in a case of adolescent anhedonia with substance misuse: Mood/theta relations," *Int. J. Psychophysiol.*, vol. 93, no. 1, pp. 84–95, 2014.

[26] D. Ariely and G. S. Berns, "Neuromarketing: The hope and hype of neuroimaging in business," *Nature Rev. Neurosci.*, vol. 11, no. 4, pp. 284–292, 2010.

[27] R. A. Ramadan and A. V. Vasilakos, "Brain computer interface: Control signals review," *Neurocomputing*, vol. 223, pp. 26–44, 2017.

[28] C. A. Kothe and S. Makeig, "BCILAB: A platform for brain–computer interface development," *J. Neural Eng.*, vol. 10, no. 5, 2013, Art. no. 056014.

[29] B. Venthur, "Design and implementation of a brain computer interface system," Ph.D dissertation, Department of Neurotechnology, Berlin Institute of Technology, Berlin, 2015. Accessed: Feb. 4, 2020. [Online]. Available: https://depositonce.tu-berlin.de/bitstream/11303/4734/2/venthur_bastian.pdf

[30] A. Delorme *et al.*, *MATLAB-Based Tools for BCI Research* (Brain-Computer Interfaces). Berlin: Germany: Springer, 2010, pp. 241–259.

[31] C. Vidaurre, T. H. Sander, and A. Schlogl, "BioSig: The free and open source software library for biomedical signal processing," *Comput. Intell. Neurosci.*, vol. 2011, 2011, Art. no. 935364.

[32] A. Schlogl and C. Brunner, "BioSig: A free and open source software library for BCI research," *Computer*, vol. 41, no. 10, pp. 44–50, 2008.

[33] P. Perego, L. Maggi, and S. Parini, "BCI++: A new framework for brain computer interface application," in *Proc. 18th Int. Conf. Softw. Eng. Data Eng.*, 2009, pp. 37–41.

[34] L. Maggi, S. Parini, P. Perego, and G. Andreoni, "BCI++: An object-oriented BCI prototyping framework," in *Proc. Int. Brain–Comput. Interface Workshop*, 2008, pp. 1–5.

[35] Y. Renard *et al.*, "Openvibe: An open-source software platform to design, test, and use brain–computer interfaces in real and virtual environments," *Presence, Teleoperators Virtual Environ.*, vol. 19, no. 1, pp. 35–53, 2010.

[36] Openvibe Group, 2019. [Online]. Available: https://gitlab.inria.fr/openvibe/sdk/

[37] J. A. Wilson, J. Mellinger, G. Schalk, and J. Williams, "A procedure for measuring latencies in brain–computer interfaces," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 7, pp. 1785–1797, Jul. 2010.

[38] I. P. Susila, S. Kanoh, K. Miyamoto, and T. Yoshinobu, "XBCI: A generic platform for development of an online BCI system," *IEEJ Trans. Elect. Electron. Eng.*, vol. 5, no. 4, pp. 467–473, 2010.

[39] B. Venthur, S. Dähne, J. Höhne, H. Heller, and B. Blankertz, "Wyrm: A brain-computer interface toolbox in Python," *Neuroinformatics*, vol. 13, no. 4, pp. 471–486, 2015.

[40] B. Venthur and B. Blankertz, "A platform-independent open-source feedback framework for BCI systems," in *Proc. 4th Int. Brain-Comput. Interface Workshop Training Course*, 2008, vol. 2008, pp. 385–389.

[41] B. Venthur *et al.*, "Pyff—A Pythonic framework for feedback applications and stimulus presentation in neuroscience," *Frontiers Neurosci.*, vol. 4, pp. 1–17, 2010.

[42] B. Venthur and B. Blankertz, "Mushu, a free-and open source BCI signal acquisition, written in Python," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2012, pp. 1786–1788.

[43] Z. Tayeb *et al.*, "Gumpy: A Python toolbox suitable for hybrid brain–computer interfaces," *J. Neural Eng.*, vol. 15, no. 6, 2018, Art. no. 065003.

[44] E. M. Mugler, C. A. Ruf, S. Halder, M. Bensch, and A. Kubler, "Design and implementation of a P300-based brain-computer interface for controlling an internet browser," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 6, pp. 599–609, Dec. 2010.

[45] M. Palankar *et al.*, "Control of a 9-DoF wheelchair-mounted robotic arm system using a P300 brain computer interface: Initial experiments," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2009, pp. 348–353.

[46] P. Brunner, S. Joshi, S. Briskin, J. R. Wolpaw, H. Bischof, and G. Schalk, "Does the 'P300' speller depend on eye gaze?" *J. Neural Eng.*, vol. 7, no. 5, 2010, Art. no. 056013.

[47] D. J. McFarland, W. A. Sarnacki, and J. R. Wolpaw, "Should the parameters of a BCI translation algorithm be continually adapted?" *J. Neurosci. Methods*, vol. 199, no. 1, pp. 103–107, 2011.

[48] P. Wei, W. He, Y. Zhou, and L. Wang, "Performance of motor imagery brain-computer interface based on anodal transcranial direct current stimulation modulation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, no. 3, pp. 404–415, May 2013.

[49] D. J. Krusienski, D. J. McFarland, and J. R. Wolpaw, "An evaluation of autoregressive spectral estimation model order for brain-computer interface applications," in *Proc. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2006, pp. 1323–1326.

[50] Y. Liu *et al.*, "Implementation of SSVEP based BCI with Emotiv EPOC," in *Proc. IEEE Int. Conf. Virtual Environ. Human-Comput. Interfaces Meas. Syst.*, 2012, pp. 34–37.

[51] E. Yin, Z. Zhou, J. Jiang, F. Chen, Y. Liu, and D. Hu, "A novel hybrid BCI speller based on the incorporation of SSVEP into the P300 paradigm," *J. Neural Eng.*, vol. 10, no. 2, 2013, Art. no. 026012.

[52] I. Volosyak, D. Valbuena, T. Luth, T. Malechka, and A. Graser, "BCI demographics II: How many (and what kinds of) people can use a high-frequency SSVEP BCI?" *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 3, pp. 232–239, Jun. 2011.

[53] Y. Höller *et al.*, "Comparison of EEG-features and classification methods for motor imagery in patients with disorders of consciousness," *PloS One*, vol. 8, no. 11, 2013, Art. no. e80479.

[54] G. Pfurtscheller *et al.*, "Walking from thought," *Brain Res.*, vol. 1071, no. 1, pp. 145–152, 2006.

[55] R. Leeb *et al.*, "Walking by thinking: The brainwaves are crucial, not the muscles!" *Presence, Teleoperators Virtual Environ.*, vol. 15, no. 5, pp. 500–514, 2006.

[56] J. Faller, G. Müller-Putz, D. Schmalstieg, and G. Pfurtscheller, "An application framework for controlling an avatar in a desktop-based virtual environment via a software SSVEP brain–computer interface," *Presence, Teleoperators Virtual Environ.*, vol. 19, no. 1, pp. 25–34, 2010.

[57] B. Z. Allison, C. Brunner, V. Kaiser, G. R. Müller-Putz, C. Neuper, and G. Pfurtscheller, "Toward a hybrid brain–computer interface based on imagined movement and visual attention," *J. Neural Eng.*, vol. 7, no. 2, 2010, Art. no. 026007.

[58] P. Perego *et al.*, "Cognitive ability assessment by brain–computer interface: Validation of a new assessment method for cognitive abilities," *J. Neurosci. Methods*, vol. 201, no. 1, pp. 239–250, 2011.

[59] M. Congedo *et al.*, "'Brain invaders': A prototype of an open-source P300-based video game working with the openvibe platform," in *Proc. 5th Int. Brain-Comput. Interface Conf.*, Sep. 2011, pp. 280–283.

[60] P. Margaux, M. Emmanuel, D. Sébastien, B. Olivier, and M. Jérémie, "Objective and subjective evaluation of online error correction during P300-based spelling," *Adv. Human-Comput. Interaction*, vol. 2012, 2012, Art. no. 4.

[61] M. Duvinage, T. Castermans, M. Petieau, T. Hoellinger, G. Cheron, and T. Dutoit, "Performance of the Emotiv EPOC headset for P300-based applications," *Biomed. Eng. Online*, vol. 12, no. 1, 2013, Art. no. 56.

[62] L. Bonnet, F. Lotte, and A. Lécuyer, "Two brains, one game: Design and evaluation of a multiuser BCI video game based on motor imagery," *IEEE Trans. Comput. Intell. AI Games*, vol. 5, no. 2, pp. 185–198, Jun. 2013.

[63] C. Zich, S. Debener, C. Kranczioch, M. G. Bleichner, I. Gutberlet, and M. De Vos, "Real-time EEG feedback during simultaneous EEG-FMRI identifies the cortical signature of motor imagery," *Neuroimage*, vol. 114, pp. 438–447, 2015.

[64] F. Lotte *et al.*, "Exploring large virtual environments by thoughts using a brain–computer interface based on motor imagery and high-level commands," *Presence, Teleoperators Virtual Environ.*, vol. 19, no. 1, pp. 54–70, 2010.

[65] I. Martišius and R. Damaševičius, "A prototype SSVEP based real time BCI gaming system," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–15, 2016.

[66] J. Legeny, R. Viciana-Abad, and A. Lecuyer, "Toward contextual SSVEP-based BCI controller: Smart activation of stimuli and control weighting," *IEEE Trans. Computat. Intell. AI Games*, vol. 5, no. 2, pp. 111–116, Jun. 2013.

[67] D. Aminaka, S. Makino, and T. M. Rutkowski, "Chromatic SSVEP BCI paradigm targeting the higher frequency EEG responses," in *Proc. IEEE Asia–Pac. Signal Inf. Process. Assoc. Annu. Summit Conf.*, 2014, pp. 1–7.

[68] S. Debener, R. Emkes, M. De Vos, and M. Bleichner, "Unobtrusive ambulatory EEG using a smartphone and flexible printed electrodes around the ear," *Sci. Rep.*, vol. 5, 2015, Art. no. 16743.

[69] M. Tariq, P. M. Trivailo, and M. Simic, "Classification of left and right knee extension motor imagery using common spatial pattern for BCI applications," *Proc. Comput. Sci.*, vol. 159, pp. 2598–2606, 2019.

[70] D. Rozado, A. Duenser, and B. Howell, "Improving the performance of an EEG-based motor imagery brain computer interface using task evoked changes in pupil diameter," *PloS One*, vol. 10, no. 3, 2015, Art. no. e0121262.

[71] N. Braun, R. Emkes, J. D. Thorne, and S. Debener, "Embodied neurofeedback with an anthropomorphic robotic hand," *Sci. Rep.*, vol. 6, 2016, Art. no. 37696.

[72] Y.-P. Lin, Y. Wang, and T.-P. Jung, "Assessing the feasibility of online SSVEP decoding in human walking using a consumer EEG headset," *J. Neuroeng. Rehabil.*, vol. 11, no. 1, pp. 1–8, 2014.

[73] S. Halder *et al.*, "Brain-controlled applications using dynamic p300 speller matrices," *Artif. Intell. Med.*, vol. 63, no. 1, pp. 7–17, 2015.

[74] A. Pinegger, H. Hiebel, S. C. Wriessnegger, and G. R. Müller-Putz, "Composing only by thought: Novel application of the P300 brain-computer interface," *PloS One*, vol. 12, no. 9, 2017, Art. no. e0181584.

[75] F. Cincotti *et al.*, "EEG-based brain-computer interface to support post-stroke motor rehabilitation of the upper limb," in *Proc. IEEE Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2012, pp. 4112–4115.

[76] R. Scherer, G. Moitzi, I. Daly, and G. R. Müller-Putz, "On the use of games for noninvasive EEG-based functional brain mapping," *IEEE Trans. Comput. Intell. AI Games*, vol. 5, no. 2, pp. 155–163, Jun. 2013.

[77] M. Wairagkar, Y. Hayashi, and S. Nasuto, "Movement intention detection from autocorrelation of EEG for BCI," in *Proc. Int. Conf. Brain Inform. Health*, 2015, pp. 212–221.

[78] M. S. Treder, N. M. Schmidt, and B. Blankertz, "Gaze-independent brain–computer interfaces based on covert attention and feature attention," *J. Neural Eng.*, vol. 8, no. 6, 2011, Art. no. 066003.

[79] S. Schaeff, M. S. Treder, B. Venthur, and B. Blankertz, "Exploring motion VEPs for gaze-independent communication," *J. Neural Eng.*, vol. 9, no. 4, 2012, Art. no. 045006.

[80] S. Bosse, M. T. Bagdasarian, W. Samek, G. Curio, and T. Wiegand, "On the stimulation frequency in SSVEP-based image quality assessment," in *Proc. 10th Int. Conf. Qual. Multimedia Experience*, 2018, pp. 1–6.

[81] TOBI Project, TiAScope, 2019. [Online]. Available: https://sourceforge.net/projects/tiascope.tools4bci.p/

[82] TOBI Project, SignalServer, 2019. [Online]. Available: https://sourceforge.net/projects/sigserver.tools4bci.p/

[83] TOBI Project, libtia, 2019. [Online]. Available: https://sourceforge.net/projects/libtia.tools4bci.p/

[84] tobicore, 2019. [Online]. Available: https://sourceforge.net/projects/tobicore.tools4bci.p/

[85] C. Breitwieser, I. Daly, C. Neuper, and G. R. Müller-Putz, "Proposing a standardized protocol for raw biosignal transmission," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 3, pp. 852–859, Mar. 2012.

[86] G. R. Müller-Putz *et al.*, "Tools for brain-computer interaction: A general concept for a hybrid BCI," *Frontiers Neuroinform.*, vol. 5, pp. 1–10, 2011.

[87] L. Sha *et al.*, "Real time scheduling theory: A historical perspective," *Real-Time Syst.*, vol. 28, no. 2/3, pp. 101–155, 2004.

[88] Inria, Jitter Monitoring and Drift Correction, 2019. [Online]. Available: http://openvibe.inria.fr/tutorial-jitter-monitoring-and-drift-correctio n/

[89] A. Schlogl, C. Brunner, R. Scherer, and A. Glatz, "BioSig: a free and open source software library for BCI research.," *Comput. 41.10*, pp. 44–50, 2008.

[90] B. Kemp, A. Vrri, A. C. Rosa, K. D. Nielsen, and J. Gade, "A simple format for exchange of digitized polygraphic recordings," *Electroencephalography Clin. Neurophysiol.*, vol. 82, no. 5, pp. 391–393, 1992.

[91] A. Schlögl, "GDF—A general dataformat for biosignals," 2006, *arXiv:cs/0608052v6*.

[92] A. Schlögl, "BioSig—An application of Octave," 2006, *arXiv:cs/0603001*.

[93] C. Kothe, "Lab streaming layer (LSL)," 2014. [Online]. Available: https://github.com/sccn/labstreaminglayer. Accessed on: Oct. 2014.

[94] C. Chen, "CiteSpace II: Detecting and visualizing emerging trends and transient patterns in scientific literature," *J. Amer. Soc. for Inf. Sci. Technol.*, vol. 57, no. 3, pp. 359–377, 2006.

[95] B. Blankertz *et al.*, "The BCI competition 2003: Progress and perspectives in detection and discrimination of EEG single trials," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1044–1051, Jun. 2004.

[96] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.

[97] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface," *J. Neural Eng.*, vol. 10, no. 4, 2013, Art. no. 046003.

[98] M. Krauledat *et al.*, "Playing pinball with non-invasive BCI," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1641–1648.

[99] A. Kübler, S. Halder, A. Furdea, and A. Hösle, "Brain painting—BCI meets art," in *Proc. 4th Int. Brain–Comput. Interface Workshop Training Course*, Graz Univ. Technol., Austria, pp. 361–366, 2008.

[100] "BCI.js," 2019. [Online]. Available: https://bci.js.org/

[101] "EEG pipes," 2019. [Online]. Available: https://github.com/neurosity/eeg-pipes

[102] "fili.js," 2019. [Online]. Available: https://github.com/markert/fili.js/

[103] "ml.js," 2019. [Online]. Available: https://github.com/mljs

[104] "Brain.js." [Online]. Available: https://github.com/BrainJS/brain.js

[105] "ConvNetJS," 2019. [Online]. Available: https://cs.stanford.edu/people/karpathy/convnetjs/

[106] C. S. Crawford and J. E. Gilbert, "Neuroblock: A block-based programming approach to neurofeedback application development," in *Proc. IEEE Symp. Visual Lang. Human-Centric Comput.*, 2017, pp. 303–307.

[107] G. Milsap, M. Collard, C. Coogan, and N. E. Crone, "BCI2000Web and WebFM: Browser-based tools for brain computer interface development and functional brain mapping," *Frontiers Neurosci.*, vol. 12, pp. 1–10, 2018.

[108] G. Placidi, A. Petracca, M. Spezialetti, and D. Iacoviello, "A modular framework for EEG web based binary brain computer interfaces to recover communication abilities in impaired people," *J. Med. Syst.*, vol. 40, no. 1, pp. 1–14, 2016.

[109] M. A. Serhani, M. El Menshawy, A. Benharref, S. Harous, and A. N. Navaz, "New algorithms for processing time-series big EEG data within mobile health monitoring systems," *Comput. Methods Programs Biomed.*, vol. 149, pp. 79–94, 2017.

[110] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to build high-performance network programs," *IEEE Internet Comput.*, vol. 14, no. 6, pp. 80–83, Nov./Dec. 2010.

[111] "LLK/scratch-vm: Virtual machine used to represent, run, and maintain the state of programs for scratch 3.0," 2019. [Online]. Available: https://www.w3.org/community/web-bluetooth/

[112] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the Web of Things," in *Proc. Internet Things*, 2010, pp. 1–8.

[113] S. Rajpoot, S. Kumar, and P. Singh, "Implementing the physical Web using bluetooth low energy based beacons and a mobile app," in *Proc. IEEE Int. Conf. Innov. Challenges Cyber Secur.*, 2016, pp. 327–329.

[114] M. Ruta, F. Scioscia, S. Ieva, G. Loseto, F. Gramegna, and A. Pinto, "Knowledge discovery and sharing in the IoT: The physical semantic web vision," in *Proc. ACM Symp. Appl. Comput.*, 2017, pp. 492–498.

[115] M. Sneps-Sneppe and D. Namiot, "On physical web models," in *Proc. IEEE Int. Siberian Conf. Control Commun.*, 2016, pp. 1–6.

[116] U. Shaked, "Reactive brain waves: How to use RxJS, angular, and web bluetooth, along with an EEG headset, to do more with your brain," 2017. [Online]. Available: https://medium.com/@urish/reactive-brain-waves-af07864bb7d4
[117] C. Escolano, A. R. Murguialday, T. Matuz, N. Birbaumer, and J. Minguez, "A telepresence robotic system operated with a P300-based brain-computer interface: Initial tests with ALS patients," in *Proc. IEEE Annu. Int. Conf. IEEE Eng. Med. Biol.*, 2010, pp. 4476–4480.
[118] L. Tonin, R. Leeb, M. Tavella, S. Perdikis, and J. d. R. Millán, "The role of shared-control in BCI-based telepresence," in *Proc. IEEE Int. Conf. Syst., Man Cybernet.*, 2010, pp. 1462–1466.
[119] L. Tonin, T. Carlson, R. Leeb, and J. d. R. Millán, "Brain-controlled telepresence robot by motor-disabled people," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2011, pp. 4227–4230.
[120] A. Chella *et al.*, "A BCI teleoperated museum robotic guide," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst.*, 2009, pp. 783–788.
[121] "Intheon neuroscale: Advanced neurotechnology, anytime, anywhere," 2019. [Online]. Available: https://neuroscale.intheon.io/
[122] Mozilla, "WebAssembly," 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/WebAssembly
[123] J. Schalkwyk, "An all-neural on-device speech recognizer," 2019. [Online]. Available: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html
[124] F. Lauer, "Mlweb: A toolkit for machine learning on the Web," *Neurocomputing*, vol. 282, pp. 74–77, 2018.
[125] D. Smilkov *et al.*, "TensorFlow.js: Machine learning for the web and beyond," 2019, *arXiv:1901.05350*.
[126] A. Saboor *et al.*, "A browser-driven SSVEP-based BCI web speller," in *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, 2018, pp. 625–630.
[127] Npm, 2019. [Online]. Available: https://www.npmjs.com/
[128] C. S. Crawford, C. Gardner-McCune, and J. E. Gilbert, "Brain-computer interface for novice programmers," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ.*, 2018, pp. 32–37.
[129] X. Hou *et al.*, "Cognimeter: EEG-based brain states monitoring," in *Transactions on Computational Science XXVIII*. Berlin, Germany: Springer, 2016, pp. 108–126.
[130] T. Hamano, T. M. Rutkowski, H. Terasawa, K. Okanoya, and K. Furukawa, "Generating an integrated musical expression with a brain-computer interface," in *Proc. New Interfaces Musical Expression*, 2013, pp. 49–54.
[131] B. Koo and S. Choi, "SSVEP response on oculus rift," in *Proc. IEEE 3rd Int. Winter Conf. Brain-Comput. Interface*, 2015, pp. 1–4.
[132] P. K. Yong and E. T. W. Ho, "Streaming brain and physiological signal acquisition system for IoT neuroscience application," in *Proc. IEEE EMBS Conf. Biomed. Eng. Sci.*, 2016, pp. 752–757.
[133] WebGL, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
[134] A. Nijholt, B. Z. Allison, and R. J. Jacob, "Brain-computer interaction: can multimodality help?" in *Proc. ACM 13th Int. Conf. Multimodal Interfaces*, 2011, pp. 35–40.