BCIManager: A library for development of brain-computer interfacing applications in Unity

Filip Škola
CYENS - Centre of Excellence
Nicosia, Cyprus
f.skola@fqs.cz

Fotis Liarokapis
CYENS - Centre of Excellence
Nicosia, Cyprus
f.liarokapis@cyens.org.cy

***Abstract*—BCIManager is a customizable library helping with the development of brain-computer interfaces (BCIs) with 3D graphics scenes, suitable for virtual reality (VR) BCI feedback, gamified BCI training, using BCIs as game inputs, and similar use cases. The library forms a layer between Unity game engine and Openvibe (software for development of BCIs) that provides control of the EEG recording process from applications made with Unity. The main feature of the BCIManager is the interface for bi-directional data exchange between Unity and Openvibe; for sending markers (stimulations) from the experimental 3D applications to the EEG recordings, and for sending classification results, features, or other data from Openvibe to the 3D applications. BCIManager also wraps around the process of starting Openvibe and can be used for arbitrary scenarios that include EEG recording with 3D/VR applications. This project is an open source, freely available at https://github.com/xskola/BCIManager.**

***Index Terms*—brain-computer interface, virtual reality, 3D graphics, games, gamification**

I. Introduction

Research in brain-computer interfaces (BCIs) attempts to create a reliable communication channel using information from the brain only [1]. Development of BCIs is motivated mainly by their ability to reconnect severely paralyzed people (with movement and communication disabilities) with the outside world [2, 3], but healthy population can also benefit from the existence of BCIs.

As the current BCIs require investment of non-trivial effort and examination of mental strategies before communication with usable accuracy is established, quality feedback from the BCI is a crucial part of the system (Figure 1 shows the typical parts of a BCI system). In recent years, studies demonstrated that scenes in 3D graphics and virtual reality (VR) are suitable for providing feedback in the BCI systems [4, 5]. This has been especially true in oscillatory paradigms [6, 7] (i.e., based on changes in neural rhythms), such as mental/motor imagery, requiring training. Motor imagery BCI trainees consciously imagine movements of their body parts, leading to modulation of neural rhythms [3, 8]. Presentation of the corresponding motor actions in 3D graphics helps in adaptation to the task [7, 9]. Furthermore, using immersive VR further helps to accelerate the learning process thanks to the body ownership transfer illusion (avatar embodiment) [7, 10].

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

Video games can be used either to mediate the training for BCIs (using gamified training [11]) or a BCI can be used as an input interface for games [12]. Gamified training can aid users to overcome psychological difficulties of the training, such as necessity to focus one's attention for prolonged periods of time, while maintain still body posture [11, 13]. Using video games can help us to understand the BCI user behavior. They can also provide an implicit performance metric and help in communicating instructions for the BCI training or usage [14].

Despite existence of game engines for fast development of advanced 3D and VR scenes (e.g., Unity and Unreal Engine) and software for development of BCI scenarios (Openvibe [15]), a tool aiding development of BCI applications in 3D or VR has been missing. BCIManager addresses this issue by providing a customizable library for bridging of the VR and the BCI development. Specifically, BCIManager helps BCI researchers and developers to use functionality of Openvibe in applications made in Unity.

Unity[1] is a game engine, freely available for non-commercial purposes and popular among researchers in various fields for its simple usage, suitable for people without deep software development skills. Openvibe[2] is a state-of-the-art open-source BCI development tool that employs an intuitive graphical language (see in Figure 2). Openvibe consists of two main components; Openvibe Acquisition Server (OVAS) ensures connection to the acquisition device (e.g., EEG), while Openvibe Designer (OVD) serves for development of the BCI components (scenarios).

BCIManager is a library that implements functionality required for the data transfer between Openvibe and Unity applications. Firstly, it provides TCP connection for a) sending event markers (stimulations) from Unity to Openvibe and b) for retrieval of the BCI data (signals, classification results, or other data) from Openvibe to Unity. Secondly, it wraps around Openvibe, taking care of starting and activating the required OVD scenario from experimental applications made in Unity.

II. Methods

From the high-level point of view, BCIManager operates Openvibe scenarios from within Unity applications. Consequently, it consists of set of sources for Unity (written in

[1]https://unity3d.com
[2]http://openvibe.inria.fr

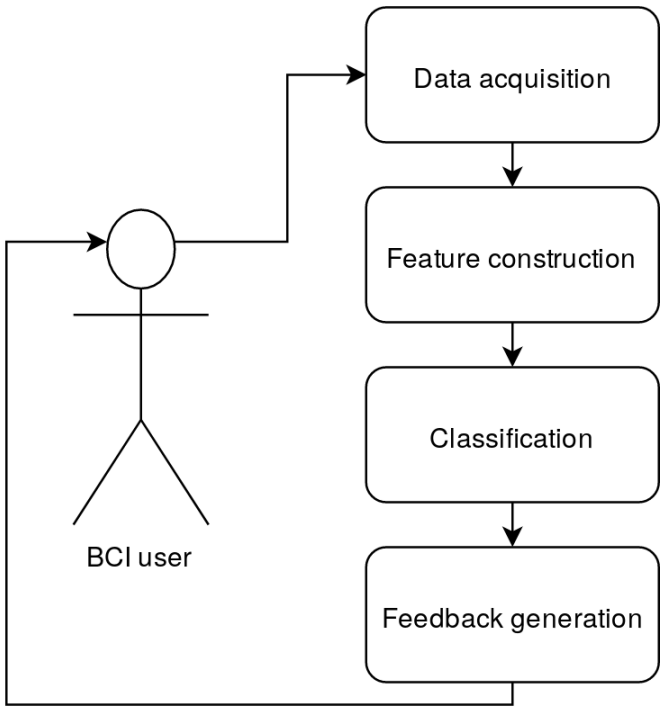

Fig. 1. Diagram of a typical BCI system. After acquisition of signals using EEG or other suitable neuroimaging system, the raw data are cleaned and processed into features. Real-time classification of the features then serves for deciding the desired action performed with a BCI. Feedback is crucial especially in the user training, as trainees adjust they mental strategies based mainly on the feedback.

C#), communicating with Openvibe scenarios (XML files), using Windows batch files as intermediates. Both Openvibe and Unity must be present on the computer, while Openvibe can optionally be present on a different machine.

The main features of BCIManager are:

- Starting OVD process with the desired scenario
- Sending event markers (stimulations)
- Receiving data (e.g., real-time classification results)
- Logging with autosave functionality
- Shutdown in case of Unity–Openvibe connection issues

A. General

To start using BCIManager in a Unity application, the main script file *BCIManager.cs* must be attached to a game object in the Unity application. *Logger.cs* should be attached as well.

Enabling/disabling features and configuration of the details for connection to OVAS and OVD is performed by changing control variables on the top of the main script *BCIManager.cs*. Mainly, users are required to check the installation path of Openvibe and the scenario they desire to run upon starting the Unity application.

The connection between Unity and Openvibe is realized using TCP/IP protocol for both directions of the data stream (from Unity to OVAS for sending stimulations, and from OVD to Unity for retrieval of BCI data).

B. Starting OVD with a scenario

The execution of OVD application is mediated using a batch file located in *OpenvibeScripts* directory (path to the batch file must be correctly set in *OPENVIBE_RUNNER_PATH* variable). The actual Openvibe scenarios (XML files) are present also in *OpenvibeScripts*, where simple scenarios for recording data, signal monitoring, receiving data, etc. are prepared. Users not experienced with development using Openvibe can start with these scenarios, or go on with modification of example scenarios provided by Openvibe installation package.

Starting of the Openvibe process by BCIManager can be disabled in the main script by setting *DO_NOT_START_OPENVIBE* to *true*. Then the developed Unity application just connects to an already running instance of Openvibe (this option is mainly present for development and debugging purposes). The variable *QUIT_ON_CONNECTION_ERROR* (set to *true* by default) takes care of early shutdown in case the connection was not established, helping to prevent experimental data loss. All actions are logged to the debug window in Unity and to external file generated by *Logger* component of BCIManager.

C. Sending stimulations

Sending stimulations to OVAS makes use of the software TCP tagging functionality of Openvibe; ”a software tagging mechanism for time-accurate placement of event markers (called stimulations in OpenViBE) in the EEG signal recordings” [16]. Sent markers are present in the data flow in OVD (from “Acquisition Client” box), available for record.

Port for connection to OVAS is found in the configuration of OVAS (“TCP tagging port”, default value 15361). Sending stimulations in BCIManager is currently implemented using *SendStimulation(stimulationCode)* function. Stimulation codes are arbitrary integer numbers, but Openvibe implements a list of commonly used stimulation codes. Handful of these (e.g., experiment start, experiment stop, start of trial, etc.) are present in *OpenvibeStimCodes.cs* file and can optionally be used instead of user-defined codes.

D. Receiving data

This feature aims to facilitate the interaction process with a BCI system. Arbitrary continuous data produced in the scope of an Openvibe scenario can be transferred to a Unity application via the TCP stream. That means that raw data can be transmitted in case this is desired, but more likely specific features or real-time classification results will be transferred to Unity and used for control of game objects.

To make the connection to OVD possible, a box called “TCP Writer” must be present in the scenario. This box starts a TCP server on a port specified in its settings (the default value is 5678) that the Unity application can connect to with BCIManager.

In BCIManager, it is required to set *RECEIVE_DATA* variable to *true* to make BCIManager try to connect to OVD. The process of receiving data is demonstrated in one of the provided example scripts.

E. Logging

Simple logging feature is part of BCIManager library, taking care of saving information from the BCI session to an external CSV file. By attaching *Logger.cs* script to a game object, logging is enabled. Apart from logging of BCIManager events, user event can be logged using *Logger.LogEvent(description)* function (description is a string containing logged event). For each line of the log, date and time, and total run time of the application are logged together with the event. Logging feature flushes new lines to disk every 2 seconds by default, thus logs are saved even if the application ends unexpectedly.

III. Use cases

Typical usage of BCIManager is with a Unity application that either records data from EEG (or other electrophysiological recording system supported by OVAS) and uses them in the real-time, or just stores them. BCIManager does not manage the OVAS component of Openvibe, as it provides connection to the acquisition device, and requires a manual set-up by experimenter. On the other hand, the OVD component is started automatically by BCIManager from the Unity application, including activation of the provided scenario. Repeated usage of an application written with BCIManager does not require experimenter to re-start other components than the experimental application (OVD is started and shut down together with the application, OVAS is running and does not disconnect from the acquisition device).

For development with BCIManager without the necessity to have the BCI acquisition device at hand, it is possible to use synthetic data. Either "Generic oscillator" or other generator of data can be used as the driver in OVAS, or "Generic stream reader" box (in OVD) can be used to read pre-recorded data from a file instead of using "Acquisition client" that reads the data from OVAS.

A. Data recording scenario

The simplest scenario for BCIManager is recording of physiological data alongside the running Unity application. In this case, a simple Openvibe scenario that records the data is sufficient (provided in example OVD scenarios). No further modifications to *BCIManager.cs* are required. Researchers utilizing data recording scenario will probably require sending stimulations marking at least the start of the visual presentation in the Unity application, and more likely multiple stimulations will be required. Those can be sent from anywhere in the Unity application by calling *BCIManager.SendStimulation(stimulationNumber)* with a stimulation number of choice as a parameter. See *ExampleScript.cs* in *BCIScripts* directory for more information on data recording scenario.

This use case was realized using development versions of BCIManager in several EEG studies [17, 18], including an ERP study of visual stimuli in VR [19].

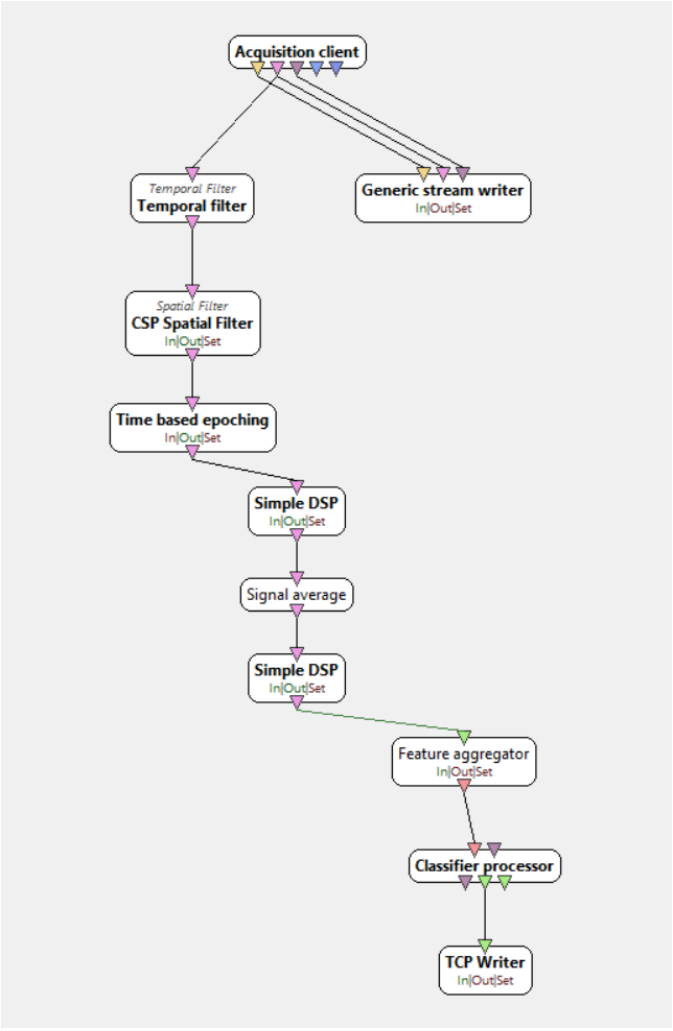

Fig. 2. Modification of the prepared Openvibe scenario for on-line BCI communication with the motor imagery paradigm. Instead of displaying the classification results in the built-in Openvibe Graz visualisation window, they are forwarded to the TCP Writer box. Using BCIManager connected to the box, real-time classification results can be used to control the interaction in the Unity application.

B. Brain-computer interaction scenario

For communication from Openvibe to Unity, data retrieval from the OVD must be enabled in *BCIManager.cs*. Developer of such a scenario typically prepares OVD files that cover most parts of the BCI system (data acquisition, processing, and classification), although it is possible to implement any part of the BCI pipeline in the Unity part of the application.

In case motor imagery paradigm is exploited for communication with a BCI, researchers can start with modifications of the tutorial Openvibe scenarios. After some data are acquired and classifiers are trained, by simple modification of *mi-csp-4-online.xml*, classifier decisions can be provided into Unity applications from the running scenario with "TCP Writer" box (displayed in Figure 2. This box can receive inputs from "Classification Processor" box, exposing hyperplane distance for each of the classes on the TCP server (in case the linear

discriminant analysis classifier is used). By subtraction of the values for each of the two classes, a simple metric about the currently predominant class is received. This metric can be improved by adding information about the probability from the classification result.

This use case was utilized in a study of gamified training for motor imagery BCI in VR [11]. Users were trained with a gamified procedure, where they controlled weapons of a space shuttle and tried to defend an Earth-like planet from asteroids (using their mental commands).

IV. Conclusions and Future Work

This paper presented BCIManager, an open source library bridging the development of BCIs in Openvibe with the development of applications for 3D graphics, VR, and games (in Unity). BCIManager covers bi-directional data flow between applications made in Unity and BCI systems written in Openvibe, helps with automatizing the recording process, and also contains a simple logging facility and data loss prevention mechanisms.

By using BCIManager, researchers in psychology, neuroscience, games, VR, and other fields can easily connect experimental applications with 3D graphics to BCI software without having to take care of networking and managing the recording software. This library is open source and freely available for researchers and developers. When using BCIManager in research, please cite this paper for reference.

In the future, we plan to extend the library with functionality specific to development of applications for mental imagery BCI paradigm. Specifically, features for coupling of organization and timing of the stimuli (e.g., for the training process) with events in Unity applications will be provided.

References

[1] Bernhard Graimann, Brendan Allison, and Gert Pfurtscheller, eds. *Brain-computer interfaces: revolutionizing human-computer interaction*. Frontiers collection. OCLC: ocn707710772. Heidelberg: Springer, 2010.

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

[3] Fabien Lotte, Laurent Bougrain, and Maureen Clerc. "Electroencephalography (EEG)-Based Brain–Computer Interfaces". In: *Wiley Encyclopedia of Electrical and Electronics Engineering* (2015).

[4] T. Sollfrank et al. "The effect of multimodal and enriched feedback on SMR-BCI performance". English. In: *Clinical Neurophysiology* 127.1 (Jan. 2016), pp. 490–498.

[5] Athanasios Vourvopoulos et al. "Effects of a Brain-Computer Interface With Virtual Reality (VR) Neurofeedback: A Pilot Study in Chronic Stroke Patients". English. In: *Frontiers in Human Neuroscience* 13 (2019).

[6] Toshiyuki Kondo et al. "Effect of instructive visual stimuli on neurofeedback training for motor imagery-based brain–computer interface". In: *Human movement science* 43 (2015), pp. 239–249.

[7] Filip Škola and Fotis Liarokapis. "Embodied VR environment facilitates motor imagery brain–computer interface training". In: *Computers & Graphics* 75 (Oct. 2018), pp. 59–71.

[8] Gert Pfurtscheller and Christa Neuper. "Motor imagery and direct brain-computer communication". In: *Proceedings of the IEEE* 89.7 (2001), pp. 1123–1134.

[9] R. Hari et al. "Activation of human primary motor cortex during action observation: A neuromagnetic study". In: *Proceedings of the National Academy of Sciences of the United States of America* 95.25 (1998-12-08), pp. 15061–15065. (Visited on 02/22/2017).

[10] Maryam Alimardani, Shuichi Nishio, and Hiroshi Ishiguro. "The Importance of Visual Feedback Design in BCIs; from Embodiment to Motor Imagery Learning". In: *PloS one* 11.9 (2016), e0161945.

[11] Filip Škola, Simona Tinková, and Fotis Liarokapis. "Progressive Training for Motor Imagery Brain-Computer Interfaces Using Gamification and Virtual Reality Embodiment". English. In: *Frontiers in Human Neuroscience* 13 (Sept. 2019), p. 329.

[12] Bojan Kerous, Filip Skola, and Fotis Liarokapis. "EEG-based BCI and video games: a progress report". English. In: *Virtual Reality* (Oct. 2017).

[13] Eva Maria Hammer et al. "Psychological predictors of SMR-BCI performance". In: *Biological Psychology* 89.1 (Jan. 2012), pp. 80–86.

[14] David A Washburn. "The games psychologists play (and the data they provide)". In: *Behavior Research Methods, Instruments, & Computers* 35.2 (2003), pp. 185–193.

[15] Yann Renard et al. "Openvibe: An open-source software platform to design, test, and use brain–computer interfaces in real and virtual environments". In: *Presence: teleoperators and virtual environments* 19.1 (2010), pp. 35–53.

[16] Nathanael Foy. *Extensions: TCP Tagging (Software Tagging)*. English. 2018. URL: http://openvibe.inria.fr/tcp-tagging/ (visited on 05/28/2021).

[17] Filip Škola and Fotis Liarokapis. "Study of full-body virtual embodiment using noninvasive brain stimulation and imaging". In: *International Journal of Human–Computer Interaction* (2021), pp. 1–14.

[18] Filip Škola et al. "Virtual Reality with 360-Video Storytelling in Cultural Heritage: Study of Presence, Engagement, and Immersion". In: *Sensors* 20.20 (2020), p. 5851.

[19] Filip Škola and Fotis Liarokapis. "Examining and Enhancing the Illusory Touch Perception in Virtual Reality Using Non-Invasive Brain Stimulation". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. New York, NY, USA: ACM, 2019, 247:1–247:12.