Interpretating EEG signals using sonification

Artur Back de Luca backdeluca.1900870@studenti.uniroma1.it
Ionut Marian Motoi motoi.1892355@studenti.uniroma1.it
Leonardo Saraceni saraceni.1920088@studenti.uniroma1.it

Abstract

We describe the task of turning Electroencephalogram (EEG) recordings into audio signals. The work here outlined concentrates on the data collection provided by Appelhoff et al. (2018)[1], mainly detecting and sonifying motor and electrooculogram (EOG) artifacts. This project is part of the Neuroengineering course hosted at the Sapienza University of Rome, and advised by Prof. Febo Cincotti.

1. Motivation

This project aims to transform incoming signals from EEG recordings into sound. We designed these sounds so to reflect the presence and/or intensity of pre-selected signatures. The resulting sounds can be used in BCIs (Brain-Computer Interfaces) as auditory feedback for any particular activity recorded by the EEG.

Another application consists of warning technicians of unwanted artifacts during a session. An EEG intend to capture physiological signals mainly produced in the brain, however this procedure is oftentimes prone to noise, potentially assigned to many sources, some somatic or external to the subject. When not addressed, these artifacts may overcome the underlying meaningful signal, either deteriorating or even invalidating a given trial. In these cases, particular sound patterns could quickly indicate a problem during a trial rather than constantly surveilling all EEG leads, which could allow to fix the source of artifacts avoiding in this way the necessity to remove faulty epochs during post-mortem data processing.

2. Project description

The task of sonifying a signal can be essentially separated into two modalities: online and offline. For these, we start with a preliminary analysis to help identifying target events and patterns to be automatically detected and later sonified. More specifically, we identify potential artifacts in the recordings, as well as event-related changes in the potential introduced by the task described in Appelhoff et al. (2018).

The second and third part are dedicated to the transformation of these signatures into sounds and their employment in a real application, respectively. As for the later, we employ the signature detection and sonification for the two said modalities. The source code was written in `python` and tested on versions 3.6 through 3.8. The application was roughly tested

1. Available at osf.io/cj2dr/wiki/home/

in different operating systems and it is compatible with Windows and Linux, although their variations were not thoroughly evaluated[2].

3. Implementation details

In the following sections we go over some of the details of the preliminary analysis of the data set, their results and implications on designing the algorithms for detection and sonification. As mentioned, we use the data set in Appelhoff et al. (2018). Since the sampling rate of the recordings is considerably high (5000 Hz), these are downsampled to 200 Hz for all the following analysis and considerations.

3.1 Feature detection

Identify EOG artifacts, bad channels

For this task we first manually search for any potential artifacts and channels that are overwhelmed by noise. Here we expose some of the diagnoses:

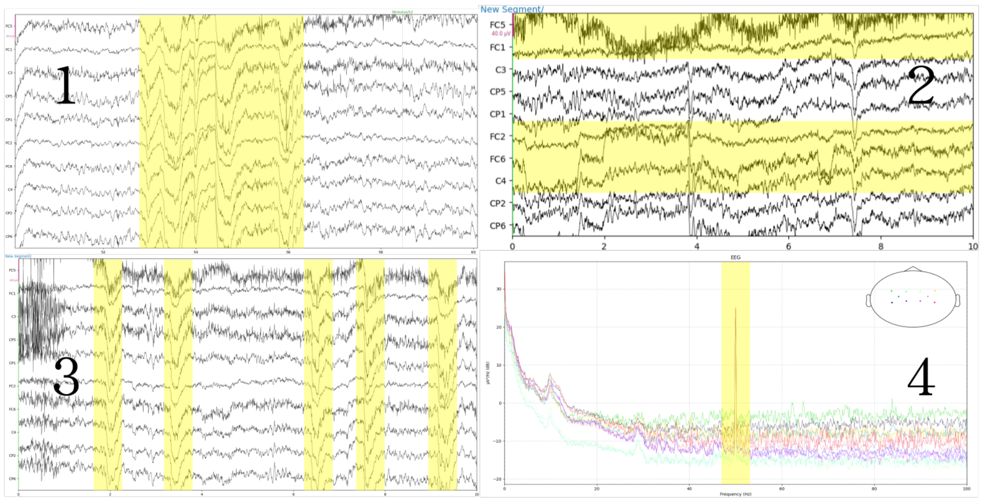

Figure 1: Artifacts manually detected in Appelhoff et al. (2018)

In this visualization we observe a plethora of artifacts, such as electrode movement (1), poor electrode placement and insulation (2) eye-related artifacts, such as blinking (3), electrical power-line interference (4). We also reckon that the channel `FC5` in (2) could be potentially discarded if not remedied by filtering techniques.

For the automatic detection of these artifacts, we first turn our attention on event-based artifacts, focusing on problems as the one shown in (1) and (3). For this first task, we follow a procedure described in Devuyst et al. (2009), utilizing a threshold based on extracted

2. For more details on the source code, refer to the project's repository: github.com/artur-deluca/eeg_sonify

statistics of the EEG signal of interest. Those values are computed independently for each lead. The formula that describes the threshold t is:

$$t = \bar{x} \pm ks, \tag{1}$$

where $\bar{x}$ and s are the mean and standard deviation of the signal's amplitude, mediated by an arbitrary multiplicative factor k. For calculating these, we load the data collection into an analytical framework named MNE, then later converting these into *numpy* arrays for the calculation of such measures. In the offline case, given the greater stability of the calculations, we verify that $k = 2$ fairly approximates the results carried out by manual inspection. However, for the online setting, since these statistics are more unstable and regularly updated, we show that $k = 4$ achieves outstanding results.

Furthermore, for the detection of persistent artifacts, not necessarily concentrated in particular time instants, we devise a band-pass filter to select a region of interest. For EOG artifacts, we select the frequency range of 1-10 Hz. With this filtered output, we can modulate the signal with respect to its frequency, amplitude or phase in order to obtain the desired sonified feedback.

Detect task related changes of SMR

The second task consisted of the detection of the sensory motor rhythm (SMR). For this, we apply a band-pass filter around 8-13 Hz, designed to only extract our region of interest: the mu-rhythm. For the ERD/ERS (Event Related Desychronization/Synchronization) computation, we select channels `CP5` and `CP6`, which have shown more significant response to stimuli. These results are then transformed into an array to be sonified in the following task.

3.2 Mapping events and states to sounds

The second task involved mapping the artifacts detected in aim 1a together with the mu rhythm extracted in aim 1b. To this end, it was observed that the best channels for this task were the frontal central channels 5 and 6, for what concerns the detection of EOG artifacts and the central parietal channels 5 and 6 for the mu rhythm observation. Other channels were viable as well but those were the ones with the strongest signal for the relative tasks. First we sonified the EOG artifacts. The way the algorithm works is by prompting the user for the channel to sonify, then for that channel it checks each sample of the artifact array obtained during aim 1a. If that sample has a value different from 0 it means that it contains an artifact at that time frame and so for the next 0.2 seconds it creates a sine wave at 440 Hz, which is an A4 note. After that, we sonified the mu rhythm that has been isolated in the previous aim. Also in this case the user is prompted for the channel he wants to sonify. This task was more problematic since the rhythm varies repeatedly during a short time. We thought that varying the intensity of a 659.25 Hz (E5 note) sine wave would be the least unpleasant for the user. A general problem common to both of the sonifications is that we had to copy each sample several times in order to match the frequency of the audio signal (44100 Hz). For the EOG artifacts sonification this had to be done only in the regions where artifacts were present, while in the second case, the one concerning the mu rhythm sonification, it was done for the whole signal. At the end, once the two sonifications are extracted, those are merged together to give an audio signal, that is the combination of the two. Therefore, in the final soundtrack, it is possible to listen to a continuous sound, that corresponds to the mu rhythm, and some peaks, that represent the artifacts detected.

3.3 Simulate live sonification of EEG

For the live sonification, we extend this procedure but solely detecting event-related artifacts for the time being. In this setting, latency becomes a great challenge since one must receive small chunks of data, calculate the necessary parameters and only then perform inference. Hence, simple and computationally efficient techniques are key.

To simulate a live recording scenario we use the LabStreamingLayer library for python (`pylsl`) throughout. In the delivering end, we first read the `.xdf` files in chunks and feed these into output outlets. As mentioned, since these recordings have a very high sampling rate, the files are considerably large and we devise and undersampling layer below the output outlets. This way from all the recorded samples in a given second, we only feed part of these so to match the new desired sampling frequency. In this stage, we can also modify other parameters, like the number of times a streaming can loop, for instance.

In the opposite end of the pipeline, in the receptive stage, we create a system that searches and fetches data from any active outlets. In our setting, we have two output streams: one for the EEG signals and other for the markers that indicate a certain event. In this section of the pipeline, we can also resize the chunks of data received and sequentially analyzed. In our context, a chunk size of approximately 250 ms achieved the best results. However, this depends on the type of problem and the size of these windows may have a direct effect on the stability of the artifact detection algorithm: small windows may generate fickle measurements and thus unreliable results. To partially remedy this, we introduce a buffer structure, that collects a predetermined number of the most recent data samples to identify the artifacts. As new data comes in, it is first examined for artifacts - if given enough data - only then to be added to the buffer. This is also the reason why for the first seconds of recording the detector remains idle, as it only starts working once it has enough data.

However, with these configurations, the number of detected artifacts remains unrealistic. Since the data chunks to be examined were considerably small, a very effective solution was to limit the appearance of a single artifact per window. This way, when detecting multiple artifacts in a window, we attain to a single event, with onset equivalent to the median of all detected artifacts. Additionally, different from the offline setting, we attempt to detect artifacts in all channels.

For this experiment it is imperative to have visual cues to compare the result of the sonification to the observable signals. To this effect, the first impulse was to use a top-shelf system such as the BrainVision LSL visualizer. However, this system is incompatible with Linux, so the solution was to come up with our own visualization scheme, mainly using `pyqtgraph`, also compatible with Windows. In this live visualization we change the signals' offset and we mark the detected artifacts as events so to enhance the diagnosis of artifacts.

In the sonification steps, the chunks where an artifact has been detected receive a single pulse with attack and release controlled by a Gaussian window, whose result is carried out by the library `sounddevice`. Since its communication with the sound driver is asynchronous, that enables the live stream of sounds without any additional delays to the setting. Additionally, in this context, the user can also save the overall audio recording to a file.

4. Final considerations

Since the sonification is entirely dependent of the quality of detection. Considering the simplicity of the techniques employed, the results are very satisfactory.

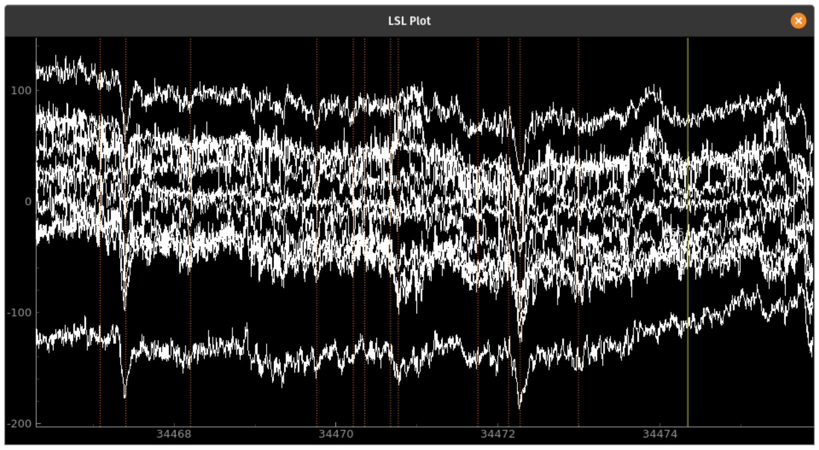

Figure 2: Final result of online artifact detection

In figure 2, the red-dotted vertical lines represent the detected artifacts, while the yellow line indicate an event. We see that the detection closely matches any visual inspection. As for sonification, we also notice that the yielded sounds show very negligible latency.

For further work, the authors recommend revisiting the mu rhythm modulation to be reliably implemented in a boxlike online setting. Additionally, other EEG signatures could be added to the overall sound such as the power-line interference, as long as their implementation do to drain extensive computational resources, which would cause latency.

In conclusion, this project has touched many distinct areas connecting concepts of statistics, signal processing, and neuroscience. Overall this was a great opportunity to exercise teamwork, learn new techniques, and practice some of the concepts learned in class.

5. Contributions

- Artur Back de Luca: live streaming and sonification. Contributed manual artifact detection and data preprocessing.
- Ionut Marian Motoi: preprocessing and offline sonification. Contributed to filtering, extraction of mu rhythm and checking the extracted artifacts by visual inspection.
- Leonardo Saraceni: artifact detection, filtering, extraction of mu rhythm and ERDS. Contributed to offline sonification.

References

Stefan Appelhoff, Daryl Sauer, and Suleman Gill. Matching pennies: A brain computer interface implementation dataset. 2018. doi: 10.17605/OSF.IO/CJ2DR. URL `https://osf.io/cj2dr/`.

S. Devuyst, Thierry Dutoit, Thierry Ravet, Patricia Stenuit, M. Kerkhofs, and E. Stanus. *Automatic Processing of EEG-EOG-EMG Artifacts in Sleep Stage Classification*, volume 23, pages 146–150. 01 2009. doi: 10.1007/978-3-540-92841-6_36.