Multimedia **A**uthoring and **M**anagement using your **E**yes and **M**ind

H2020-ICT-2014 - 644780

D2.3

Final integration and optimization of multi-modal sensors

Dissemination level:	Public (PU)
Contractual date of delivery:	Month 32, 31/12/2017
Actual date of delivery:	Month 32, 08/01/2018
Workpackage:	WP2 Sensor Configuration and Signal Capturing
Task:	T2.1, T2.2, T2.3, T2.4
Type:	Prototype
Approval Status:	Final
Version:	1.2
Number of pages:	57
Filename:	D2.3_Final_integration_and_optimization_of_multimodal_sensors_Final.pdf

Abstract: Deliverable D2.3 deals with the final implementation of the sensor modules of WP2 and provides suggestions for the best practices in case of drawbacks during future data acquisition. The data collected in the Phase I dataset have been exploited to carry out an analysis and refinement of the filtering algorithms; no particular issues were found related to the acquisition of the raw data.

co-funded by the European Union

Copyright

History

Version	Date	Reason	Revised by
0.1	12/10/17	Table of Contents	Dario Comanducci
1.0	27/11/2017	Beta version ready for reviewers	Dario Comanducci
1.1 (pre-final)	14/12/2017	Beta version after reviews	Dario Comanducci Elisavet Chatzilari Korok Sengupta
1.2 (final)	08/01/2018	Final version	Dario Comanducci

Author list

Organization	Name	Contact Information
EBN	Dario Comanducci	dario.comanducci@ebneuro.com
EBN	Paolo Giovagnola	paolo.giovagnola@ebneuro.com
EBN	Francesco Ferrara	ferrara@ebneuro.com
EBN	Michele Venturi	venturi@ebneuro.com
EBN	Edoardo Sabatini	edoardo.sabatini@ebenuro.com
CERTH	Elisavet Chatzilari	ehatzi@iti.gr
CERTH	Fotis Kalaganis	fkalaganis@iti.gr
CERTH	Georgios Liaros	geoliaros@iti.gr
CERTH	Katerina Adam	katadam@iti.gr
CERTH	Spiros Nikolopoulos	nikolopo@iti.gr
CERTH	Ioannis (Yiannis) Kompatsiaris	ikom@iti.gr
UNI KO-LD	Raphael Menges	raphaelmenges@uni-koblenz.de
UNI KO-LD	Chandan Kumar	kumar@uni-koblenz.de
UNI KO-LD	Steffen Staab	staab@uni-koblenz.de

Executive Summary

The content of the current deliverable deals with the final implementation of the sensor modules of WP2 and provides suggestions for the best practices in case of drawbacks during future data acquisition. The data collected in the Phase I dataset have been exploited to carry out an analysis and refinement of the filtering algorithms; no particular issues were found related to the acquisition of the raw data.

Abbreviations and Acronyms

API	Application Programming Interface
BCI	Brain Computer Interface
CAR	Common Average Reference
ECG	ElectroCardioGram
ET	Eye tracker
EEG	ElectroEncephaloGram
GSR	Galvanic Skin Response
MD	Muscular Disorder
PD	Parkinson Disease
HR	Heart Rate

Contents

List of Figures

List of Tables

1 WP2 goals and achievements

The general objective of WP2 is to provide guidelines and best practices for the configuration, calibration and synchronization of the sensor modules that constitute the core of MAMEM technology. Furthermore, WP2 deals with the acquisition and preprocessing (filtering) of the raw signals acquired by the sensors.

These activities have been undertaken in three separate tasks, one for each sensor module:

- T2.1 – Configuration, calibration and noise-reduction in eye tracking movements
- T2.2 – Configuration, calibration and noise-reduction in EEG signals
- T2.3 – Configuration, calibration and noise-reduction in bio-measurements

Finally, task T2.4 had the role to optimize and integrate the previous activities. Tab. 1 summarizes the results achieved before starting the ending part of WP2.

Tab. 1: Results achieved for WP2 before the ending part.

Commitments	Progress
Device selection based on project requirements	Defined sensors for both Heavyweight and Lightweight configuration (D2.1 [5] and D4.2 [8])
Configuration and Calibration	All configuration and calibration procedure defined and described (D2.1 [5] and D4.2 [8])
Signal Synchronization and data acquisition	Sensors to middleware wrappers implemented and tested, synchronization procedure defined and tested successfully with minor issues on EEG Lightweight headset fixed (D2.2 [6])
Noise and artifact reduction	Bio signal recorder and ET filter algorithms implemented (D3.1 [7]). EEG filters implemented on Matlab (D2.1 [5], D2.2 [6]), then ported to C/C++
Near real time optimization	Trials of Phase I was completed without relevant issues, in terms of signal acquisition (only some drawbacks due to heavyweight eyetracker overheating)

The content of the current deliverable deals with the final implementation of the sensor modules and provides suggestions for the best practices in case of drawbacks during future data acquisition.

From the trials acquired for the Phase I dataset [26] no issues were found concerning the low-level modules implemented so far. Therefore the last part of WP2 mainly concerned the improvement of the signal quality. To accomplish this, improvements were made both in terms of hardware and software.

Regarding the hardware, in order to obtain EEG signals with a quality better than the Emotiv Epoc headset (used as lightweight device in the trials of Phase I dataset), ENOBIO 8 has been selected as new lightweight EEG headset to be used in the final phase of the project. The features of ENOBIO 8 are detailed in Section 3.1.

Concerning the software, the data collected in the dataset have been used to improve the filtering algorithms of the data used in MAMEM project (eye-tracker data, EEG signals and Galvanic Skin Response). Moreover, upon the

experience gained by the inspection of Phase I dataset, a list of best practices and recommendations have been identified and summarized in the following of the deliverable: Section 2 concerns the final tuning of eye tracker module; Section 3 deals with the EEG module, and finally Section 5 is about GSR data.

Section 4 contains a new kind of analysis on EEG data, that is more advanced than "simple" filter preprocessing. The aim of this study is to devise a methodology that can measure the quality of the acquired EEG signals. This quality measure could be very useful in future developments: a continuous bad quality index found on a specific channel may rise an online warning that the corresponding electrode needs an inspection (i.e. the index can substitute in a first run the standard ohmmeter check procedure described in deliverable D4.2 to asses the quality of the EEG data to be acquired). Also in this case, the trials of Phase I dataset have been used in the tuning and evaluation of the proposed method. Although this task is more suitable for the Middleware layer of WP4 (because it is more than raw data filtering of WP2 and less than classification algorithms in WP3, applied to the EEG potentials related to the use of the system), the study is described here for convenience of writing and because of the connections with the noise to be removed.

2 Final tuning about eye tracker data acquisition and best practices

Several factors influence gaze estimation of remote eye tracking systems. Users may move their head, change their visual angle in relation to the system or wear visual aid that distorts the recorded image and leads to problems in the estimation of gaze. Furthermore, there are environmental influences as ambient lighting or the hardware itself bears limitations like camera sensor resolution or the implemented calibration algorithm. These factors lead to problems both in accuracy and precision. Accuracy, also referred to as *Gaze Drift*, describes the average distance of gaze estimation to the true user fixation. Precision, also referred to as *Gaze Jitter*, describes the spread of gaze samples for a fixation (see Fig. 1).

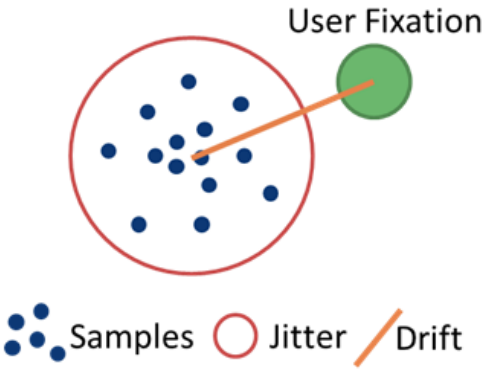


Fig. 1: Visual explanation for Gaze Drift (accuracy) and Gaze Jitter (precision).

2.1 Improving precision

To improve precision, we apply a combination of weighted smoothing of gaze samples over a sliding time window, a saccade detection and finally an outlier correction [14].

2.1.1 Weighted average

A basic attempt to smooth gaze data is to collect samples over time and to accumulate their coordinates. This can be performed for a sliding time window, to cover e.g., the current user fixation but not the whole gaze history. By declaring a gaze sample at the current time t as $\mathbf{x}_t$, N as window size and w as applied weight per sample, we can define the following general formula to compute the weighted average of gaze $\hat{\mathbf{x}}_t$:

$$\hat{\mathbf{x}}_t = \sum_{i=0}^{N-1} \frac{w_i}{\sum_j w_j} \mathbf{x}_{t-i} \tag{1}$$

The weight w can be defined by an additional kernel function, which takes the age of the sample into account. Literature names three common kernels to

calculate a weight per sample in the window:

Linear Kernel $w_i = 1$. Every sample is weighted equally

Triangular Kernel $w_i = N - i$. Newest sample is weighted the highest, oldest with 1.

Gaussian Kernel $w_i = \mathrm{e}^{-\frac{i^2}{2\sigma}}$, where $\sigma = \sqrt{\frac{-(N-1)^2}{2\ln 0.05}}$, assign the oldest sample in the window a weight of 0.05.

While all three kernels have been integrated to GazeTheWeb, the best results are to be expected from the Gaussian kernel [14]: hence, it has been chosen as weighting kernel for the second trial phase.

2.1.2 Saccade detection

The described approach is based on the assumption that the gaze samples inside the sliding time window belong to a single fixation. This is not the case when the user performs a saccade. Then the window might contain samples from two or more fixations. Averaging these samples would produce a fixation somewhere in the middle between the contained fixations, possibly on a region which the user never fixated. Therefore, the sliding window for filtering is limited to the current fixation by using a distance threshold to distinguish between gaze samples belonging to the current fixation or another one. The current fixation is then defined as gaze samples which distance is successive below this threshold, starting from the latest.

As GazeTheWeb is rendered 60 times a second on the screen, in every frame the available gaze samples are iterated from newest to oldest, until the threshold is exceeded by the spatial distance of two successive sample. The collected samples are filtered and used for further input handling.

2.1.3 Outlier correction

Eye tracking may produce single outliers, e.g., when a reflection is on the camera or the data transfer suffers of an error. This may produce a single outlying gaze sample, which would prohibit a proper filtering of a fixation as described before. Therefore, when going from the newest to the oldest sample within the sliding window, for each sample that is classified to belong potentially to another fixation, the previous and next sample are checked to belong to the same fixation according to the spatial distance of the coordinates and the threshold. If this is the case, the currently classified sample is discarded for the filter process as outlier and the filtering is continued. Otherwise the sample collection is stopped and the weighted average of the collected samples is calculated.

2.2 Improving accuracy

To improve accuracy, we offer users a hands-free activation of recalibration and we will evaluate the benefits of a novel implicit online drift correction, which is based on user interactions.

2.2.1 Calibration Process

For the second trial phase, myGaze eye trackers will be utilized for gaze estimation. We call the calibration procedure provided by the accompanying API via `iV_Calibrate()`, in order to setup the system for usage. The API provides a 5-point-calibration, where the gaze of the user must follow a dot on the screen. The dot pauses on five distinct positions, for which the fixation data is collected automatically. For each calibration point, the quality of the acquired data can be checked. In case of failure a new attempt is initiated through calling the function `iV_RecalibrateOnePoint(pointNumber)`.

Through changes in environmental conditions or user positioning, the calibration may break during usage and the accuracy of gaze estimation decreases. In order to provide to motor-impaired users an accessible way to trigger a recalibration in addition to the initial calibration after application start-up, the user can close their eyes for a predefined number of seconds. After this time, an alert is emitted and the screen changes, as depicted in Fig. 2. The user is presented two dwell time based buttons. One small one on the left side to continue with browsing and a big one on the right side to start recalibration. The size of the button is sufficient to trigger recalibration even for a very bad calibration.

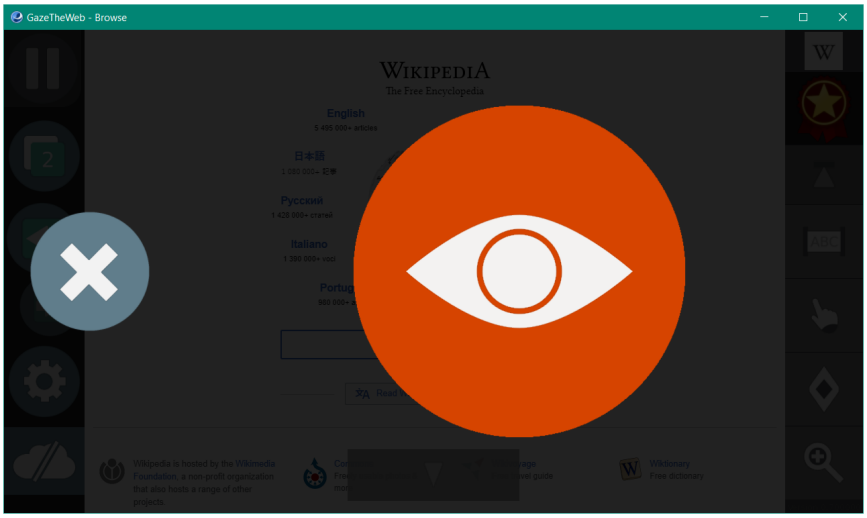


Fig. 2: Screen for recalibration.

2.2.2 Drift map

In addition to the explicit calibration of the eye tracking environment through the myGaze software, we plan to perform an implicit on-top calibration with a drift map. For this approach we assume the user to look at the center of buttons while dwelling them for activation, e.g., the back button in the browsing interface or the keys of the text input screen. For each successful activation of such a button we measure the distance between center of the button and the estimated and filtered gaze coordinate of the user. This drift data is accumulated in a grid. The data from the grid can be utilized to offset the estimated gaze and eventually to improve the accuracy of the system.

As this is a novel idea, we plan to activate and deactivate this feature on daily base without further notification of the users. Since the approach is not directly visible to the user, we will receive unbiased feedback about the success of this method by measuring the count of recalibrations and the general usage data. The presence or absence of this feature will still allow the smooth interaction of the participants and is not expected to cause any frustration to them, rather an improvement of interaction with the activated system compared to the baseline.

2.3 Best practices

The myGaze eye tracking system works similarly to the heavy-weight SMI RED-n with dark-pupil tracking. The eyes of the users are lit from both far ends of the device. The camera sensor is placed in the center of the device. It records the the eyes of the user which show reflections of the light sources. Both the dark pupil and the reflections are detected on the image and used to estimate the gaze. The systems lacks an additional centered light source next to the camera sensor, which allows for a direct corneal reflection. This reflection would result in a bright pupil in the recorded image, which is more robust to detect in case of an externally caused reflection. Therefore, positioning the system nearby a window or other light source should be avoided for a sufficient interaction performance of the setup.

3 Final tuning about EEG data acquisition and best practices

During the creation of Phase I dataset no particular acquisition issues due to the low level modules of the EEG devices were found. In any case, in order to get a better EEG signal quality for the lightweight system, another device has been considered as new EEG headset to be used during the Phase II trials: the Neuroelectrics ENOBIO 8 [25], that will substitutes the Emotiv Epoc previously discussed in D2.1 and D2.2. ENOBIO 8 is already compatible with the Labstreaminglayer standard, that represents the low-level interface of the Middleware, thus allowing a fast integration in the MAMEM platform; see Section 3.1 for a detailed discussion of the features of the new lightweight headset.

Phase I dataset [26] (available at [11]) has been used also as a real benchmark for the filtering algorithms proposed in D2.1, since this dataset can be regarded much closer to the actual scenarios of Phase II compared to the SS-VEP datasets (available at [9] and [10]) previously utilized in the analysis described in D2.2. In particular, the combination of spatial filters such as CAR (Common Average Reference) and Amuse has been investigated in Section 3.2, in order to find the final recommendations for their future use. In D2.2 we compared several ways to fix the independent components (ICs) computed by Amuse and the correction of noisy ICs based on wavelet analysis was found to be the best ("WT cor" in Section 4 of D2.2). Here only that method is further analyzed; in the following it will be labeled with the more clear name "Wavelet-Amuse".

Moreover, a novel algorithm based on robust estimation techniques, has been proposed to remove the power-line interference in challenging conditions; Section 3.3 explains in detail the theory behind the new algorithm and provides an experimental evaluation of its effectiveness. A simplified version of the proposed method has been integrated also in EB Neuro's software products for the analisys of EEG Evoked Potentials.

3.1 ENOBIO 8 as new lightweight headset

ENOBIO 8 is a wearable, wireless, portable electrophysiology sensor system for the recording of electroencephalogram (EEG). Over 70 publications during the last 10 years have been made possible using ENOBIO, which adds to the credibility of the device. It was preferred to the previously selected Emotiv EPOC+ device, due to the higher quality of the captured signals and its modularity (i.e. the position and the type of the electrodes is selected by the user, see Fig. 3).

ENOBIO 8 offers 8 channels plus 2 for reference for capturing the EEG signals. Apart from EEG, ENOBIO 8 is capable to record also electrooculogram (EOG), electrocardiogram (ECG) and electromyogram (EMG). According to our experience, ENOBIO 8 proved to be an invaluable asset for out-of-the-lab experiments. The convenient use, modern and appealing look as well as its almost no preparation time required is a key aspect for the MAMEM project and its end target group of impaired people. ENOBIO 8 also allows offline data storage by providing direct access to microSD card. With respect to software, ENOBIO 8 is accompanied, at no additional fee, with NIC (the Neuroelectrics Instrument Controller engine) that is in charge for controlling all Enobio systems. NIC provides real time access to physiological signal data (quality check,

Fig. 3: ENOBIO 8 headset and its electrodes.

raw data, filtered data, spectral features, scalp maps, etc.) and is available for both OS X (Mac) and Windows operating systems.

Finally, the greatest advantage ENOBIO 8 has to offer is the flexible electrode placement with 39 possible positions based on the 10-10 system. Additionally it allows alternating between wet, dry and solid-gel (disposable pads that are placed between the electrode and the head to increase conductivity) electrodes with no effort as long as one owns the different types of electrodes that can be obtained with a small additional cost. We must note here that ENOBIO 8 is a CE certified medical device suitable for electrophysiological research, medical application development and brain computer interfaces. The detailed specifications of ENOBIO 8 device are outlined in Tab. 2.

Tab. 2: Neuroelectrics ENOBIO 8 specifications.

Specification	Value
Number of Channels	8 Channels, flexible placement
Reference Electrodes	In the CMS/DRL noise cancellation configuration mastoids location
Bandwidth	0 to 125 Hz (DC coupled)
Sampling rate	500 SPS
Resolution	24 bits – 0,05 microvolt (μV)
Measurement Noise	< 1 μV RMS
Input impedance	1 GΩ
Communication	Bluetooth 3.0 and 2.1
Output	EDF+, ASCII, NEDF data files or TCP/IP raw data streaming
Storage	MicroSD card for on-board storage
Battery	USB rechargeable system using Li-Ion battery
Battery Duration	Operating time up to 16 hours
Dimensions	$60 \times 85 \times 20$ mm
Weight	65 g
Lab Streaming Layer	Direct Support
Certificate	93/42/CEE Medical Devices Directive (as amended by 2007/47/CE Medical Devices Directive).

3.2 Evaluation of CAR/Amuse in noisy scenarios

The EEG data of Phase I dataset have been utilized as a benchmark to refine the parameters and find the final recommendations for the spatial filters CAR and Wavelet-Amuse.The dataset contains several trials performed both on patients and healthy subjects. The trials are divided into three cohorts accordingly to the center (and related disability) where they have been recorded: AUTH cohort (Parkinson disease), MDA cohort (neuromuscular disease) and SHEBA cohort (spinal chord injury).

As a first analysis, the EEG data have been imported in the EEG Galileo software (the commercial software developed by EB Neuro [13]) to carry out a visual inspection of the signals and the effects of the EEG filtering process. After the selection of EEG data chunks with challenging (but reasonable) noise, both from healthy subjects and patients, the CAR and wavelet-Amuse filters[1] have been evaluated: in order to accomplish this the EB Neuro team has inserted an *ad-hoc* call to the EEG filter library inside the processing pipeline code of Galileo. Section 3.2.1 shows qualitatively the performance that the two filters can achieve and their limits.

Then, a second analysis has evaluated the effects of the Amuse filter related to the the classification of ErrPs potentials (in Section 3.2.2) and provides some quantitative data about that.

3.2.1 Visual evaluation

Ocular artifacts The first example in Figs. 4–6 concerns ocular artifacts. Fig. 4 reports the unprocessed signal, while Fig. 5 illustrates the effect of the CAR filter only: the filter reduces the artifacts in most of the channel, but not in the frontal ones. Only the subsequent application of wavelet-Amuse removes the ocular artifacts, by fixing the first five independent components (Fig. 6).

Channel highly disturbed In Fig. 7 channel O2 (plotted in red) is highly disturbed, and the CAR filter cannot remove such an artifact (Fig. 8) because it is not a common mode noise. Wavelet-Amuse must fix up to the first eight components, but its effects is to change also channel Ft8 (cyan channel in Fig. 9); furthermore, the frontal electrodes (top channels) remains very noisy.

Strong power-line interference Channels C3, P1, Pz, POz of Fig. 10 (plotted in red) have a strong interference at 50 and 100 Hz that comes from the power supply (In Fig. 10 the standard notch filter at 50 and 100 Hz is off to graphically emphasize the interference). Fig. 11 shows the effect of CAR filter: although in this case the notch filter is on, the interference is so high that it cannot completely be removed and it is propagated to the other channels by the CAR filter. Neither Wavelet-Amuse can remove such an interference (Fig. 12), but at least it does not propagate it to the other channels.

Miscellaneous noise Fig. 13 shows a combination of several artifacts: muscular and ocular artifacts, and a drift in a channel. Wavelet-Amuse can cope with two of these disturbs: the ocular artifacts and drift, while muscular artifacts remain (Fig. 14). An explanation of this fact will be provided later in Section 3.4.

[1] see the filter library reported in [5].

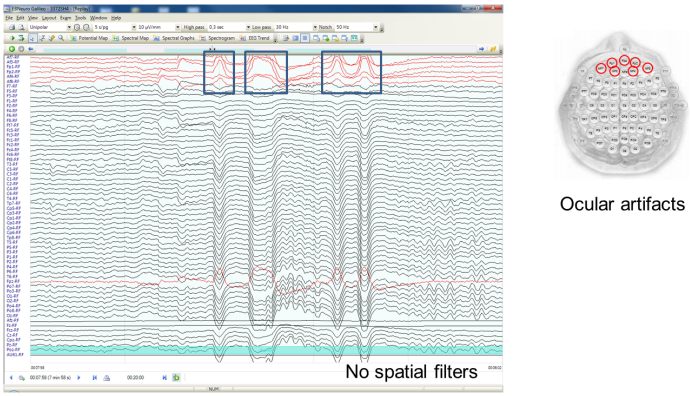


Fig. 4: Example of ocular artifacts (from healthy subject no. 4 of Sheba cohort, SMR experiment with the heavyweight configuration).

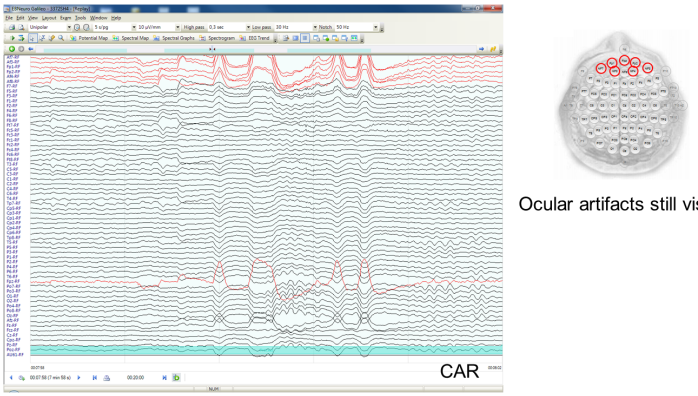


Fig. 5: CAR filter reduce ocular artifacts in most of the channel but not in the frontal ones.

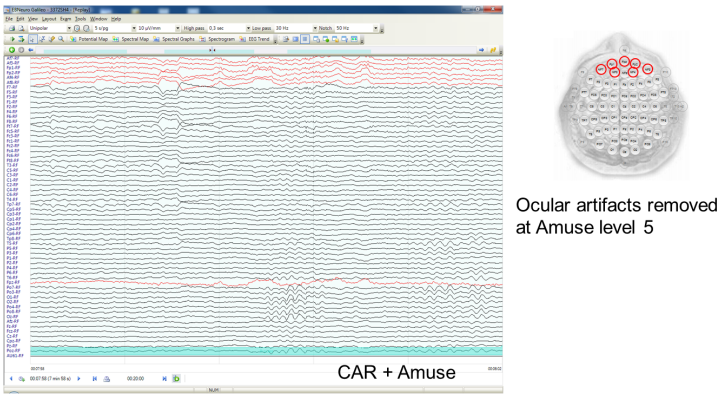


Fig. 6: EEG data after fixing the first 5 components of Wavelet-Amuse decomposition.

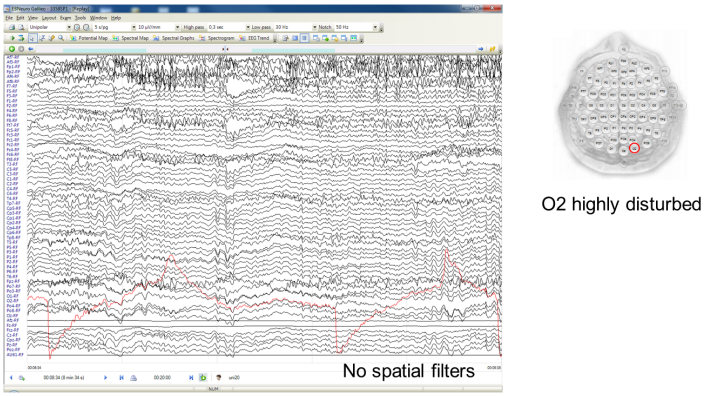


Fig. 7: Highly disturbed data for channel O2, in red (patient no. 1 of SHEBA cohort, SMR experiment with the heavyweight configuration).

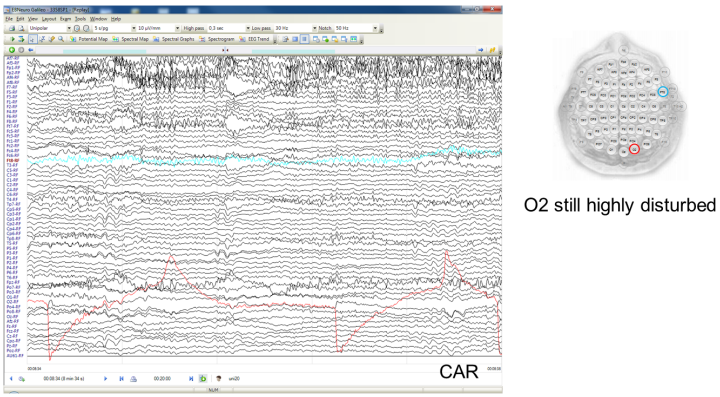


Fig. 8: CAR filter can not remove the noise from channel O2.

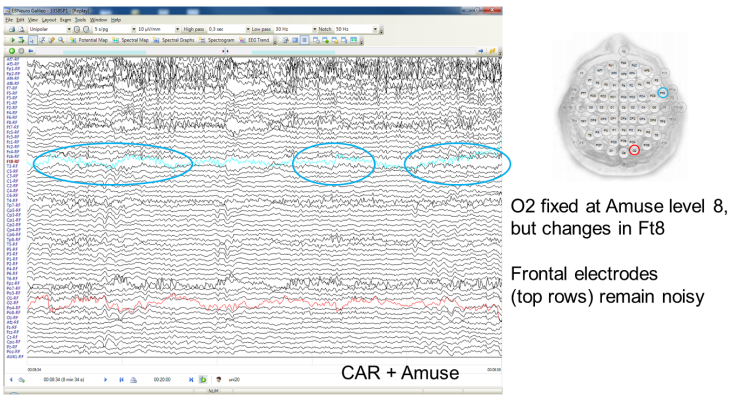


Fig. 9: To remove the artifacts in channel O2, Wavelet-Amuse must fix the first eight components, but its effect changes significantly also channel Ft8.

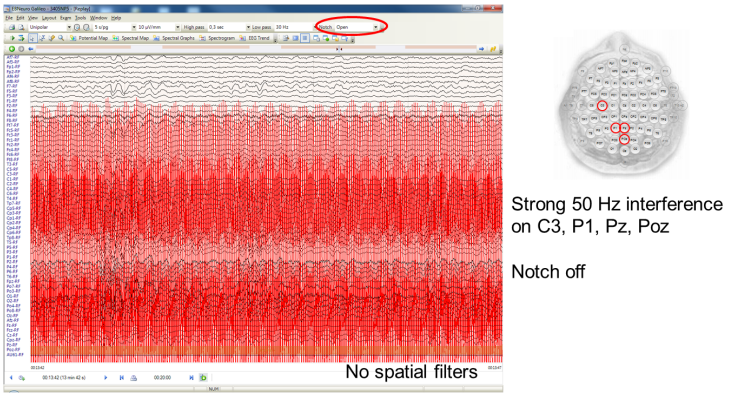


Fig. 10: Strong power-line interference for channels C3, P1, Pz ans POz, in red (from patient no. 5 of MDA cohort, SMR experiment with the heavyweight configuration).

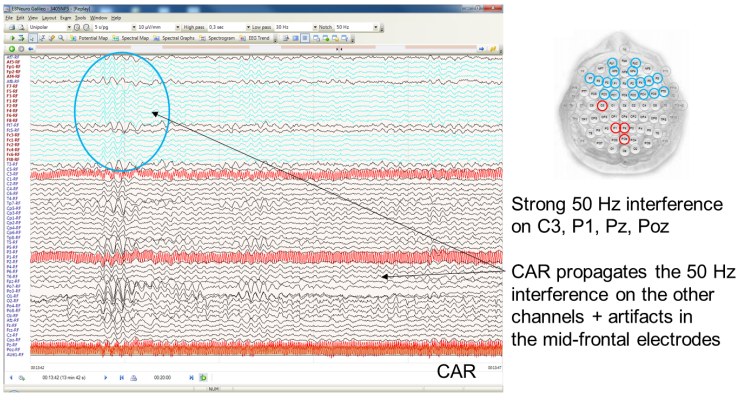


Fig. 11: CAR filter propagates the interference collected by electrodes C3, P1, Pz, POz to the other channels. Moreover, some artifacts are now visible in the mid-frontal electrodes (in cyan).

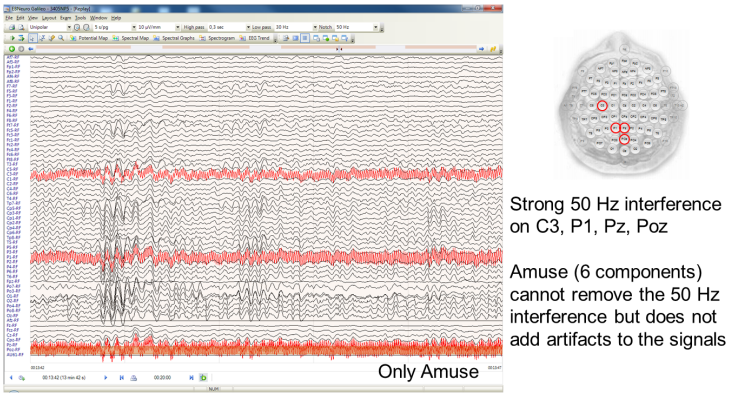


Fig. 12: Wavelet-Amuse (working on the first six components) is not able to remove the power-line interference, but at least it does not propagate it to the other channels.

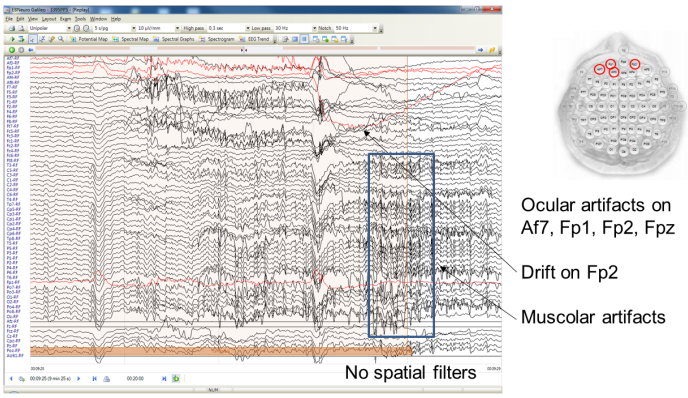


Fig. 13: Ocular artifacts in Af7, Fp1, Fp2, and Fpz; drift on Fp2, and several muscular artifacts (from patient no. 5 of AUTH cohort, SMR experiment with the heavyweight configuration).

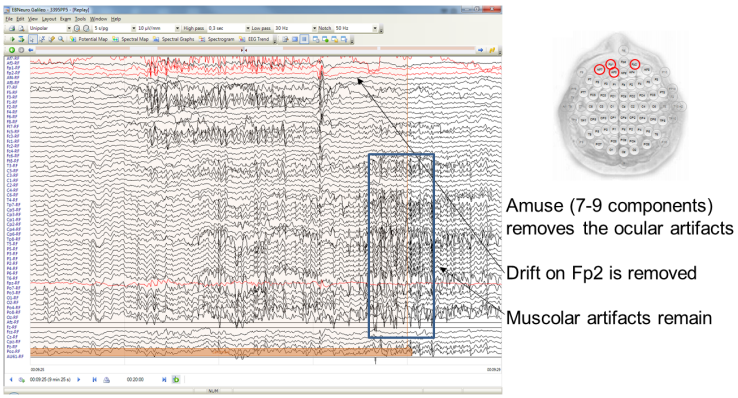


Fig. 14: Wavelet-Amuse (working on the first seven-nine components) removes the ocular artifacts and the drift on Fp2, but not the muscular artifacts.

Lightweight configuration So far, the examples come from acquisition with the heavyweight configuration, but similar situations hold for the lightweight system: as an example, Figs. 15 and 16 show that Wavelet-Amuse still removes the ocular artifacts but not the other noise when it works with few (14) channels.

3.2.2 Wavelet-Amuse and ErrP classification performance

One of the aim of MAMEM project is the use of EEG signals in conjunction with an eye-tracker in order to create a high-speed gaze-based keyboard.

This task is described in detail in deliverable D3.3 (to be published), where it will be demonstrated that a specific neurophysiological event associated with error perception is elicited during the visual mistyping perception, which is not contaminated by eye movement artifacts. This event, which is a special case of an ErrP, can serve as the basis for an automated Error Correction System that originates directly from the users' brain responses complemented by the eye movement information and is able to improve the overall typing speed in

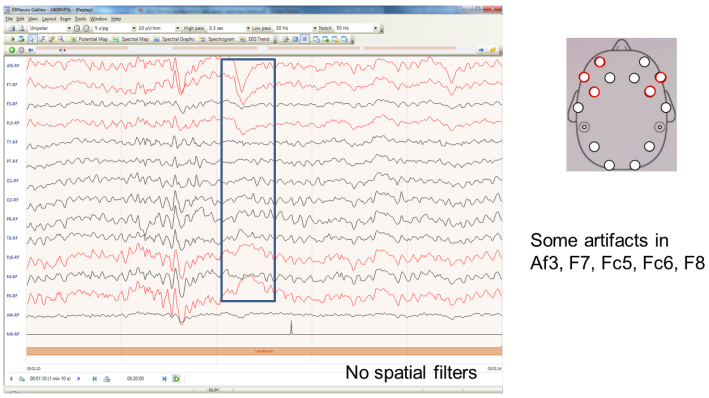


Fig. 15: Ocular artifacts (inside the blue rectangle) in Af3, F7, Fc5, Fc6, F8, plus other spread noise (from patient no. 5 of MDA. cohort, SMR experiment with the lightweight configuration).

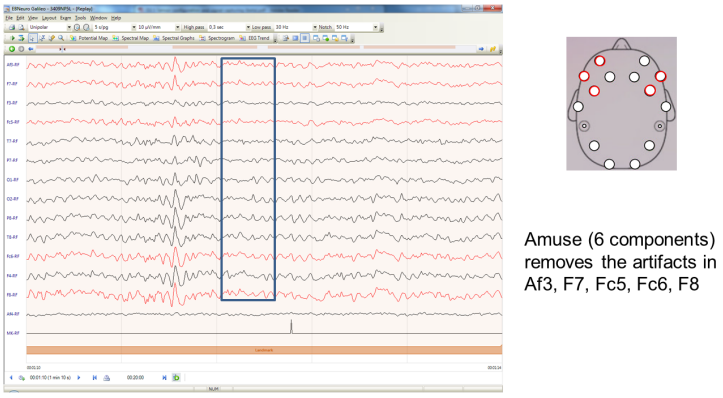


Fig. 16: Wavelet-Amuse (working on the first six components) removes the ocular artifacts from the area in the blue rectangle (compare with Fig. 15), but not the spread noise.

gaze-based visual keyboard.

All the volunteers that joined the experiments (that will be released through D3.3) had no known prior or current pathological neurological condition, and their vision was normal or corrected to normal. EEG and gaze data were collected by mean of the already acquisition tools developed for the MAMEM system: to capture the brain's electrical activity the heavyweight EBNeuro EEG device was utilized (sampling rate at 256 Hz), while for the gaze information the SMI myGaze eye-tracker was used (30 Hz); the devices were synchronized by the LabStreamingLayer module of the Middleware [8].

To understand if the Wavelet-Amuse processing can improve the detection of the ErrPs due to mistyping, an off-line experiment has been carried out by integrating the filter in the preprocessing steps before the classification starts.

A subset of the subjects of the D3.3 experiments was considered for this study, in function of the classification accuracy gained in their trials according

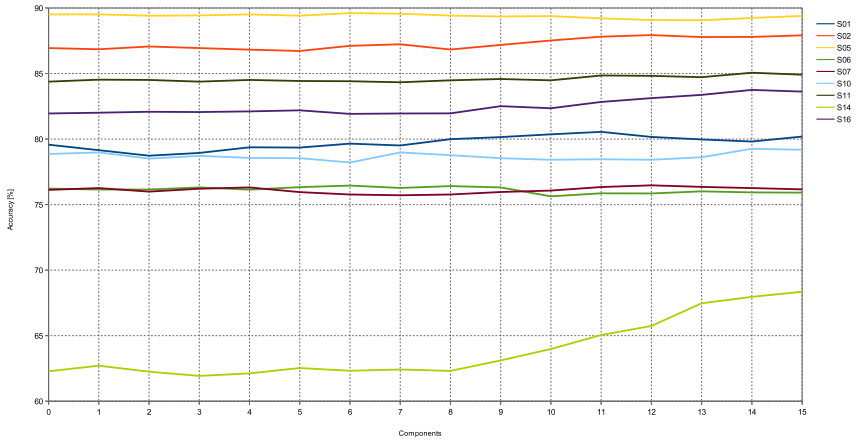


Fig. 17: Evaluation of ErrPs accuracy as function of independent components fixed by Wavelet-Amuse. For most of the subjects the accuracy does not change significantly, with the exception of the worse case that increases from 62.28% to 68.35%.

to the following criteria:

- the first two subjects with the highest accuracy (respectively 89.51% and 86.94% of accuracy), labeled in the following as *good* subjects;
- the *worse* subject (62.28%);
- among the remaining people, all with a comparable accuracy between 75-85%, a random selection of other six persons labeled as *medium* subjects (four of them are in the range 75-80% and the other two in 80-85%).

The graph in Fig. 17 shows how the classification accuracy changes as the number of components used by the wavelet correction increases. For the medium and good subjects, the scores remain more or less stable, with a slight improvement trend in some of them (e.g. subject 02 and 07). Instead the classification of the worse subject has received noticeable benefits from the Wavelet-Amuse preprocessing: from the initial value of 62.28%, the accuracy increases and reaches 68.35% with 15 components.

3.3 Notch filter as robust linear regression

Although modern biomedical amplifiers have a very high common mode rejection ratio, recordings are often contaminated by residual power-line interference. This is due to differences in the electrode impedances and to bias currents through the patient front-end. Thus, the common mode voltage is transformed into a pseudo differential signal that cannot be suppressed even by an infinitely high CMRR [21].

If the power-line waveform is not a pure sinusoid due to distortions or clipping, harmonics of the fundamental frequency could also appear [28]. When the power-line interference is very intense[2], the harmonics higher than half of

[2] The power-line interference can be higher than 1V, while the EEG potential is in the order of μV.

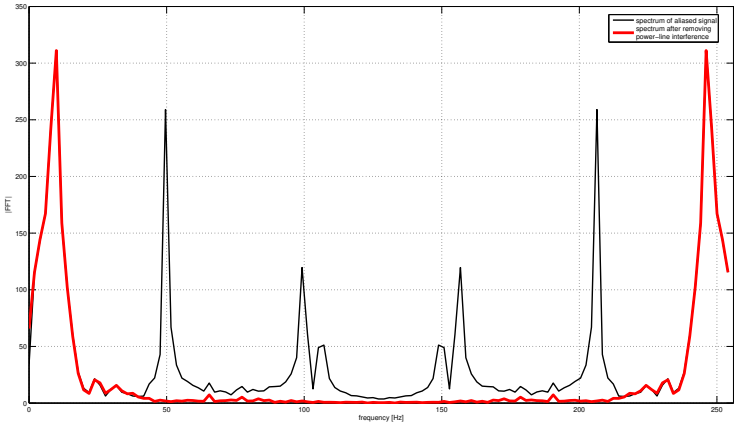


Fig. 18: In black (thin line), the spectrum of a signal where a sinusoidal component at 150 Hz has been added artificially to an EEG chunk (500 ms long) sampled at 256 Hz. The EEG chunk has a dominant frequency around 10 Hz, plus a power-line interference at 50 and 100 Hz; the artificial component at 150 Hz produces also an aliased component of the spectrum around 106 Hz. In red (bold line), the spectrum of the signal after the removal of the power-line interference by the method proposed in the following Sections: the "prohibited" component at 150 Hz is properly removed together with the aliased term at 106 Hz.

the sampling rate could not be adequately reduced by the analog antialiasing low-pass filter of the device; hence the power-line signal will be distorted and aliased by these harmonics (see Fig. 18 for an explanatory example).

Moreover, in presence of signal discontinuities (e.g. sudden spikes due to other artifacts), the standard digital notch filters have a ringing effect[3] that modify the signal waveform.

In any case, removing effectively the power-line interference is a good practise when using spatial filters: as already shown in Section. 3.2.1 about CAR filter (Fig. 12), even the power-line interference (not completely removed) of few channels may significantly waste the effect of spatial filters; this fact can be extended to the other spatial filters that perform some weighted average of the channels (e.g. Laplacian or Common Spatial Pattern filters).

Hence for all these reasons, in order for the MAMEM system work in uncontrolled environments, a different approach to the classic notch filtering has been developed to remove the power-line interference. This robust method have been added to the EEG filtering library already proposed in D2.1 [5]; for the implementation details see appendix A.

The proposed method is described as follows. First, Section 3.3.1 shows how to convert the estimation of the power-line interference into a linear regression problem. Then, in Section 3.3.2 a robust method based on M-estimators is introduced to estimate the power-line signal when a biological signal and (eventually) other artifacts/noise are superimposed to it. Finally, Sections 3.3.3 and 3.3.4 provide experimental evaluations of the proposed method.

[3] I.e. the artificial oscillations that occurs in sharp transitions or discontinuities of the data when a narrow-band filtering is performed.

3.3.1 Linear regression

Distorted power-line interference $m(t)$ due to the electrical mains is modeled by several harmonics[4] of 50 Hz (or 60 Hz, depending on the country):

$$m(t) = \sum_{\kappa=1}^{K} \alpha_\kappa \sin(2\pi f_\kappa t + \varphi_\kappa) \tag{2}$$

In general, the signal to be analyzed will be of the form

$$y(t) = b(t) + n(t) + m(t) \tag{3}$$

where $b(t)$ is the biological signal (i.e. EEG), $n(t)$ represents additional artifacts and/or random noise, and $m(t)$ is the power-line signal modeled by Eq. 2. Once estimated $m(t)$, given a sampled signal $y_i = y(t_i)$, $i = 1 \dots N$, the power-line interference can be filtered out by the subtraction $y(t) - m(t)$: the remaining signal contains only the biological signal $b(t)$ and the additional noise $n(t)$.

The estimation of a sum of K sine waves can be regarded as a linear regression problem. In fact, given the samples y_i and being $\omega_\kappa = 2\pi f_\kappa$, Eq. 2 can be written as (let us ignore for the moment $b(t)$ and $n(t)$ in Eq. 3, since they are regarded as noise for the estimation of $m(t)$):

$$\begin{aligned} y_i &= \sum_{\kappa=1}^{K} \alpha_\kappa \sin(\omega_\kappa t_i + \varphi_\kappa) \\ &= \sum_{\kappa=1}^{K} \underbrace{\alpha_\kappa \cos\varphi_\kappa}_{v_\kappa} \underbrace{\sin(\omega_\kappa t_i)}_{a_{i,k}} + \underbrace{\alpha_\kappa \sin\varphi_\kappa}_{v_{K+\kappa}} \underbrace{\cos(\omega_\kappa t_i)}_{a_{i,K+\kappa}} \\ &= \sum_{\kappa=1}^{2K} a_{i,\kappa} v_\kappa \end{aligned} \tag{4}$$

To be a bit more general in the computation, also a constant offset v_0 is introduced into the signal modeling the power-line interference:

$$y_i = v_0 + \sum_{\kappa=1}^{2K} a_{i,\kappa} v_\kappa \tag{5}$$

For each sample y_i, Eq. 5 contributes to the following system:

$$\underbrace{\begin{bmatrix} 1 & a_{1,1} & \dots & a_{1,2K} \\ 1 & a_{2,1} & \dots & a_{2,2K} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{N,1} & \dots & a_{N,2K} \end{bmatrix}}_{\mathtt{A}} \underbrace{\begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{2K} \end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}} \tag{6}$$

With $N = 2K + 1$, the system $\mathtt{A}\mathbf{v} = \mathbf{y}$ of Eq. 6 can be solved exactly; when $N > 2K + 1$, a least square solution can be obtained e.g. through normal equations[5] by exploiting $\mathtt{A}^+$, the pseudo-inverse of $\mathtt{A}$:

$$\mathbf{v} = \mathtt{A}^+\mathbf{y} = (\mathtt{A}^\top\mathtt{A})^{-1}\mathtt{A}^\top\mathbf{y} \tag{7}$$

[4] E.g. f_κ= 50, 100, 150, ... Hz.

[5] Other methods can be used to compute the least square solution; as an example, SVD decomposition is one of them.

Finally, the substitution of the values for $v_1, \ldots, v_K$ in Eq. 4 is sufficient to estimate the power-line interference at each sample y_i; in any case, it is also possible to recover the amplitude and phase of each sinusoidal wave by the following formulas:

$$\varphi_\kappa = \arctan\left(\frac{v_{K+\kappa}}{v_\kappa}\right) \tag{8}$$

$$\alpha_\kappa = \frac{v_\kappa}{\cos\varphi_\kappa} \tag{9}$$

The offset v_0 is not strictly required to remove the power-line interference.

3.3.2 Robust linear regression

Since y_i may contain also the contribution of the biological signal $b_i = b(t_i)$ and some other noise $n_i = n(t_i)$ (see Eq. 3), the standard least square approach is not suitable to recover the value of the power-line interference $m_i = m(t_i)$. Least squares are optimal in the case of Gaussian error summed to the model to be estimated, and this is not the case (even under the assumption that $n(t)$ is a Gaussian noise, $b(t)$ in general is not Gaussian).

The effect of $b(t)$ and $n(t)$ in this context can be regarded as the production, in the data collection $\{y_i\}_{i=1}^N$, of outliers[6] w.r.t. the power-line model.In the literature there have been developed a lot of robust algorithms to cope with outliers [16]: in this work, a robust technique based on M-estimators have been evaluated in recovering the true value for $m(t)$ in presence of outliers.

The standard least-squares regression method tries to minimize $\sum_i r_i^2$, being r_i the residual of the fitted model w.r.t. the corresponding datum y_i (let $\mathbf{a}_i^\top$ denotes the i-th row of matrix $\mathtt{A}$ in Eq. 6):

$$r_i = y_i - \mathbf{a}_i^\top \mathbf{v} \tag{10}$$

M-estimators form a class of robust (maximum-likelyhood) estimators that try to reduce the effect of outliers by replacing the squared residuals r_i^2 by another function $\rho(\cdot)$ of the residuals r_i, yielding

$$\min \sum_i \rho(r_i) \tag{11}$$

The *loss function* $\rho(r)$ is required to be a symmetric, positive-definite function with a unique minimum at zero, and must be *less increasing than the square function* [30].

Three M-estimators have been considered in this work, chosen among the widely used (see e.g. [15][22][30] for a review), and each of them is characterized by its loss function:

L1-L2

$$\rho_L(r) = 2(\sqrt{1 + r^2/2} - 1) \tag{12}$$

Huber ($\eta = 1.345$)

$$\rho_H(r) = \begin{cases} r^2/2 & \text{if } |r| \le \eta \\ \eta\,(|r| - \eta/2) & \text{if } |r| > \eta \end{cases} \tag{13}$$

[6] Outliers are samples that are far from the data model distribution, and they can corrupt in a hugely arbitrary way the least squares estimation.

Tukey ($\tau = 4.6851$)

$$\rho_T(r) = \begin{cases} \tau/6\,(1-(1-(r/\tau)^2)^3) & \text{if } |r| \le \tau \\ \tau/6 & \text{if } |r| > \tau \end{cases} \tag{14}$$

All these M-estimators for small residuals approximate the function $r^2/2$. For big residuals, both L1-L2 and Huber have a loss function that increases linearly (an oblique limit in case of L1-L2) while Tukey loss function becomes constant. Fig. 19 shows the plot of the loss function of the three M-estimators, together with their weight function $w(r)$, defined as:

$$w(r) = \frac{d}{dr}\rho(r) \tag{15}$$

Please note in Fig. 19(c) that large outliers are not considered by Tukey estimator, since their weights are all zeros. Analytically, the weight functions of the three M-estimators are:

L1-L2

$$w_L(r) = \frac{1}{\sqrt{1+r^2/2}} \tag{16}$$

Huber

$$w_H(r) = \begin{cases} 1 & \text{if } |r| \le \eta \\ \eta/|r| & \text{if } |r| > \eta \end{cases} \tag{17}$$

Tukey

$$w_T(r) = \begin{cases} (1-(r/\tau)^2)^2 & \text{if } |r| \le \tau \\ 0 & \text{if } |r| > \tau \end{cases} \tag{18}$$

The weight function is exploited in the Iteratively Reweighted Least Squares (IRLS) algorithm, a technique to obtain the solution for Eq. 11; for details about the origin of $w(r)$ see [15][30]. The IRLS algorithm is summarized in Alg. 1; by setting $\mathtt{W} = \mathrm{diag}(w_1, \ldots, w_N)$, the value for $\mathbf{v}$ corresponding to $\min \sum_i w_i r_i^2$ can be computed as

$$\mathbf{v} = (\mathtt{A}^\top \mathtt{W} \mathtt{A})^{-1} \mathtt{A}^\top \mathtt{W} \mathbf{y} \tag{19}$$

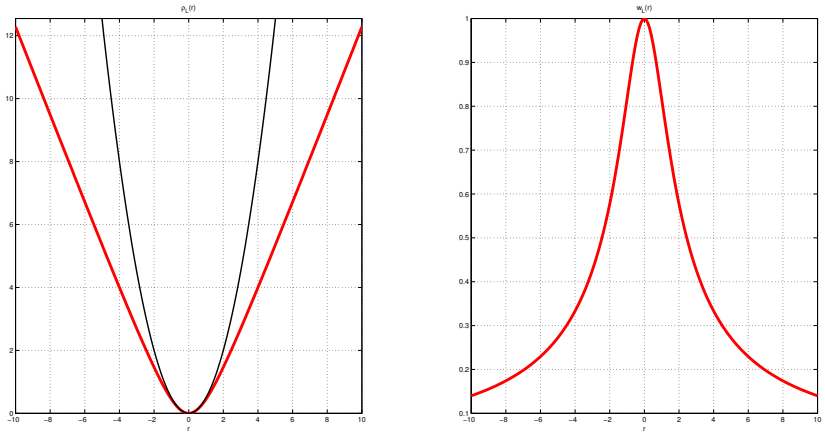
Algorithm 1: Iteratively Reweighted Least Squares procedure

Input: sample data $\{y_i\}_{i=1}^N$
Output: power-line model m_i

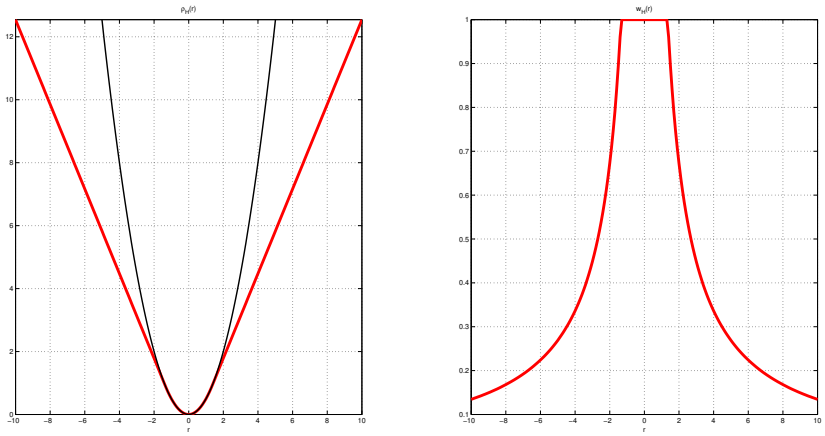
1 Obtain an initial solution $\mathbf{v}^{(0)}$ (e.g. by Least Squares) producing residuals $r_i^{(0)}$
2 At step n, computes the weights $w_i = w(r_i^{(n-1)})$
3 Solve for $\min \sum_i w_i r_i^2$ (according to Eq. 19) and goto 2 until convergence

So far, for sake of simplicity, the scale of the residuals has been neglected. However, in order to achieve scale invariance of the estimators, the Median of Absolute Deviation (MAD) [16][22]

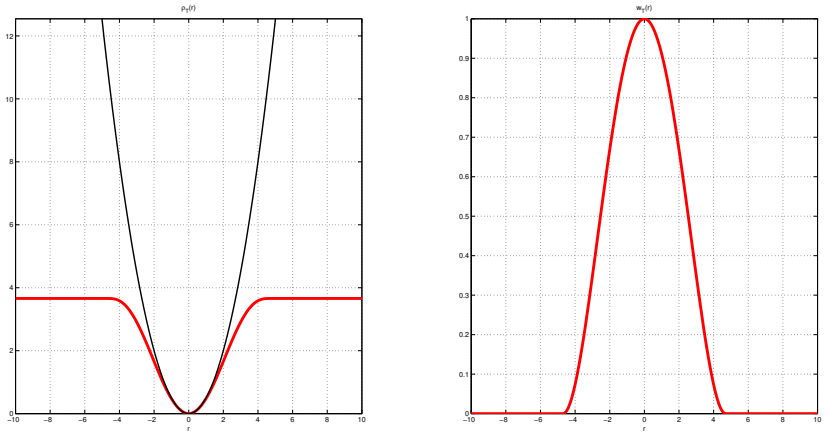
$$\sigma = \frac{\mathrm{median}(|\mathrm{r_i} - \mathrm{median}(\mathrm{r_i})|)}{0.6745} \approx 1.48\,\mathrm{median}(|\mathrm{r_i} - \mathrm{median}(\mathrm{r_i})|) \tag{20}$$



(a) L1-L2



(b) Huber



(c) Tukey

Fig. 19: Loss function $\rho(r)$ (on the left) and weight function $w(r)$ (on the right) for the three M-estimators. The function $r^2/2$ is also plotted with a thin black line in the left drawing of Figs. 19(a), 19(b) and 19(c).

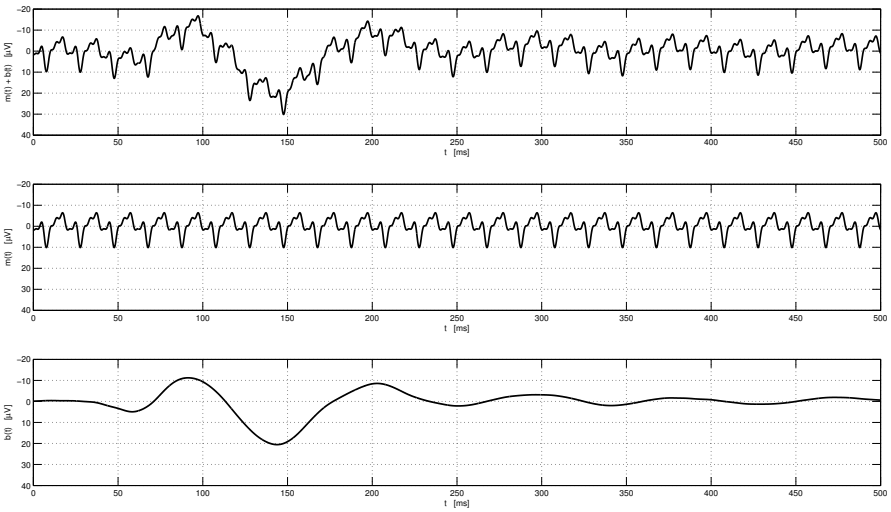


Fig. 20: Example of distorted power-line interference $m(t)$ superimposed to an EEG potential $b(t)$ (top row). The artifact may be present in all channels or in isolated channels that include electrodes that have poorly matched impedances. The middle row and bottom row show separately the power-line signal $m(t)$ and the EEG potential $b(t)$.

should be incorporated during the computation of the weights by normalizing the residuals by σ:

$$w_i = w(r_i/\sigma) \tag{21}$$

A case study To graphically explain how M-estimators work, let us use the synthetic example of Fig. 20 (top row), obtained as the superposition of ground truth signals for the distorted power-line interference $m(t)$ and the EEG potential $b(t)$; the two signals $m(t)$ and $b(t)$ are shown separately in the other two rows of the same figure (see Section. 3.3.3 for the details about $m(t)$ and $b(t)$).

Fig. 21 shows the behaviour of L1-L2 estimator, both in terms of power-line model estimation and final weight computation. The power-line interference has been perfectly recovered thanks to the weights that are reduced when the corresponding sample y_i is an outlier for the model (e.g. the samples between 100 and 200 ms). Finally, also in presence of Gaussian noise, as shown in Fig. 22, the estimation of the power-line interference is correctly computed.[7]

3.3.3 A Montecarlo experiment

To evaluate the effectiveness and robustness of the robust linear regression, a Montecarlo experiment has been carried out by summing a known synthetic power-line interference $m(t)$ to a real EEG potential $b(t)$ and an increasing Gaussian noise $n(t)$ (see Eq. 3). The experiment compares the behaviour of the three M-estimators (L1-L2, Huber and Tukey) exposed in Section 3.3.2.

[7] Huber and Tukey estimators have similar performance both for the case of Fig. 21 and Fig. 22: being the main visible difference only in the weights (they are less smooth than L1-L2, due to the hard thresholds η and τ in their weight functions of Eq. 17 and Eq. 18), the results of the two estimators have not been plotted.

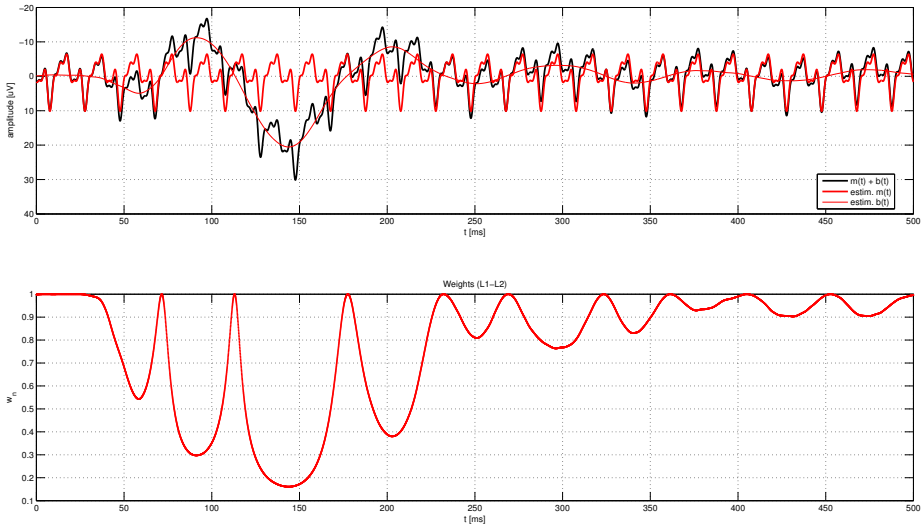


Fig. 21: The result obtained by L1-L2 estimator when applied to the synthetic example of Fig. 20.

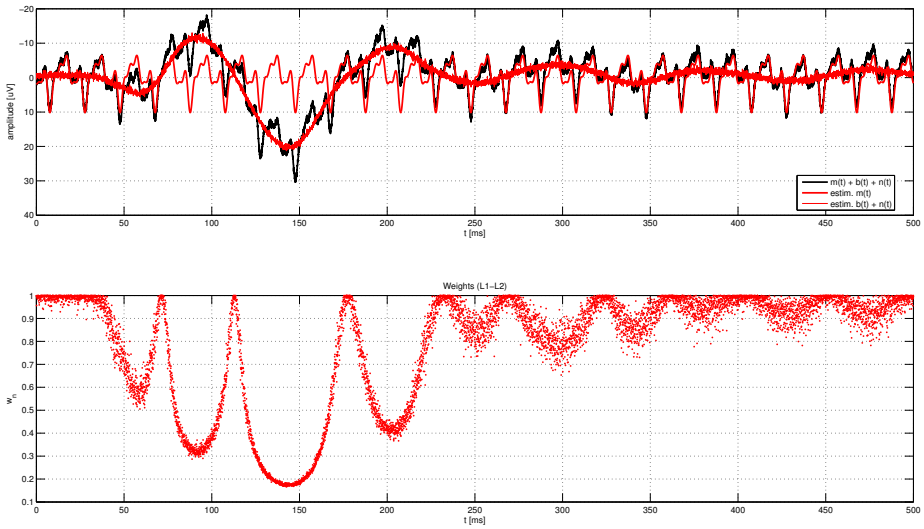


Fig. 22: The result obtained by L1-L2 in presence of additional Gaussian noise applied to the synthetic example of Fig. 20.

Tab. 3: Coefficients for the synthetic power-line interference, obtained from a real acquisition with moderate interference.

κ	Frequency f_κ [Hz]	Amplitude α_κ [μV]	Phase φ_κ [rad]
1	50	4.37	-0.49
2	100	2.23	2.06
3	150	2.54	0.59
4	200	0.82	-1.74
5	250	1.46	2.27

The signal $m(t)$ was measured during an internal trial session as power-line interference in an environment close to several electronics/electrical devices; its purpose here is to provide a ground truth signal for the following experiments. The EEG potential $b(t)$ was still obtained in an internal trial session, and here it has the main the purpose of adding to $m(t)$ a vertical offset varying in time as a disturb in the estimation of the actual $m(t)$.

In order to test the estimation of several harmonics (compliant to the Shannon Sampling Theorem), the Montecarlo experiment is here performed at a frequency of 512 Hz, thus allowing in Eq. 2 $f_\kappa = 50, 100, 150, 200, 250$ Hz. Although the trials in MAMEM are collected at 256 Hz, we just moved to the next value among the sampling rates usually exploited in EEG recordings, thus being not so far from the MAMEM standard[8] (with a sampling rate at 256 Hz, we would have only two harmonics < 128 Hz to be evaluated). The remaining coefficents of the sinusoidal functions for $m(t)$ in Eq. 2 are listed in Tab. 3.

Moreover, by multiplying the original power-line interference $m(t)$ by different amplitude scaling factors A, three scenarios have been generated:

- strong interference, with scaling factor $A = 5$
- medium interference, with scaling factor $A = 1$
- low interference, with scaling factor $A = 0.5$

Finally, for each scenario an increasing Gaussian noise $n(t)$ is added with standard deviation $\sigma_G = 0$ (no noise), 0.1, 0.5, 1.0, 1.5, 2.0, 2.5; for a given level of noise σ_G, 50000 random instances of noise $n(t)$ are added to the original signal $b(t) + A\,m(t)$.

The output results have been averaged and compared in Tab. 4 (strong interference), Tab. 5 (medium interference) and Tab. 6 (low interference).[9] The tables report the number of iterations required by each M-estimator to converge, and the distribution (summarized by the average value and standard deviation) of the estimated coefficients for the power-line interference $A\,m(t)$.

As expected from Tabs. 4–6, as the interference decreases (from strong to low) the estimation performance for $A\,m(t)$ gets worse, becoming predominant

[8] Furthermore, the new lightweight headset ENOBIO 8 has a sampling rate of 500 Hz.

[9] In Tab. 6 some entries are not available (n.a.) because the corresponding output files (each containing 50000 trials) were unreadable due to a formatting issue, and averaging the results was not possible. Since they are not expected to provide relevant information, and given the long computational time required by the Montecarlo experiment, we avoid to run again the related estimations.

Tab. 4: Results for the strong interference case. For each each M-estimator, the table reports the number of iterations required to converge, and the distribution (summarized by the average value and standard deviation) of the estimated coefficients for the power-line interference $A\,m(t)$.

Method	Noise	Iterations	50 Hz		100 Hz		150 Hz		200 Hz		250 Hz	
		#/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.
G.Truth	–	-/-	21.87 μV	-0.49 rad	11.13 μV	2.06 rad	12.69 μV	0.59 rad	4.10 μV	-1.74 rad	7.29 μV	2.27 rad
L1-L2	0 0	31/-	21.90/-	-0.49/-	11.12/-	2.06/-	12.67/-	0.59/-	4.14/-	-1.71/-	7.44/-	2.17/-
	0 1	22.04/6.67	21.90/0.01	-0.49/0.00	11.12/0.01	2.06/0.00	12.67/0.01	0.59/0.00	4.14/0.01	-1.71/0.00	7.45/0.01	2.17/0.00
	0 5	22.60/9.83	21.90/0.07	-0.49/0.00	11.12/0.07	2.06/0.01	12.67/0.07	0.59/0.01	4.13/0.07	-1.71/0.01	7.47/0.07	2.16/0.01
	1 0	22.01/8.91	21.91/0.14	-0.49/0.01	11.12/0.14	2.06/0.01	12.67/0.14	0.59/0.01	4.13/0.14	-1.72/0.03	7.50/0.14	2.14/0.02
	1 5	22.10/7.35	21.91/0.21	-0.49/0.01	11.12/0.21	2.06/0.02	12.68/0.21	0.59/0.02	4.13/0.21	-1.72/0.05	7.54/0.20	2.12/0.03
	2 0	20.40/6.32	21.91/0.27	-0.49/0.01	11.12/0.27	2.06/0.02	12.68/0.27	0.59/0.02	4.13/0.27	-1.72/0.07	7.59/0.27	2.10/0.04
	2 5	19.70/5.29	21.91/0.34	-0.49/0.02	11.11/0.34	2.06/0.03	12.68/0.34	0.59/0.02	4.13/0.33	-1.73/0.08	7.64/0.33	2.08/0.05
Huber	0 0	15/-	21.91/-	-0.49/-	11.12/-	2.06/-	12.68/-	0.59/-	4.16/-	-1.70/-	7.43/-	2.20/-
	0 1	21.77/9.80	21.91/0.01	-0.49/0.00	11.12/0.01	2.06/0.00	12.68/0.01	0.59/0.00	4.16/0.01	-1.70/0.02	7.43/0.01	2.20/0.00
	0 5	24.60/13.8	21.91/0.07	-0.49/0.00	11.12/0.07	2.06/0.01	12.68/0.07	0.59/0.01	4.16/0.07	-1.70/0.03	7.44/0.07	2.19/0.01
	1 0	24.14/13.4	21.91/0.14	-0.49/0.01	11.12/0.14	2.06/0.01	12.69/0.15	0.59/0.01	4.16/0.14	-1.71/0.05	7.46/0.14	2.17/0.02
	1 5	23.40/12.6	21.91/0.21	-0.49/0.01	11.12/0.21	2.06/0.02	12.69/0.21	0.59/0.02	4.15/0.21	-1.72/0.07	7.50/0.21	2.15/0.03
	2 0	22.54/11.4	21.91/0.28	-0.49/0.01	11.12/0.28	2.06/0.03	12.69/0.28	0.59/0.02	4.14/0.28	-1.73/0.07	7.55/0.28	2.13/0.04
	2 5	21.81/10.1	21.91/0.34	-0.49/0.02	11.12/0.34	2.06/0.03	12.69/0.34	0.59/0.03	4.13/0.34	-1.73/0.08	7.60/0.34	2.11/0.05
Tukey	0 0	20/-	21.84/-	-0.49/-	11.18/-	2.06/-	12.72/-	0.59/-	4.26/-	-1.70/-	7.25/-	2.33/-
	0 1	27.00/13.6	21.84/0.02	-0.49/0.00	11.18/0.02	2.06/0.00	12.72/0.02	0.59/0.00	4.26/0.02	-1.70/0.01	7.25/0.02	2.33/0.00
	0 5	35.36/24.0	21.87/0.08	-0.49/0.00	11.17/0.08	2.06/0.01	12.71/0.09	0.59/0.00	4.26/0.08	-1.70/0.02	7.27/0.09	2.33/0.02
	1 0	35.49/24.6	21.90/0.16	-0.49/0.01	11.16/0.16	2.06/0.02	12.70/0.17	0.59/0.01	4.22/0.17	-1.71/0.04	7.33/0.17	2.31/0.03
	1 5	33.92/23.8	21.90/0.23	-0.49/0.01	11.14/0.24	2.06/0.02	12.70/0.24	0.59/0.02	4.16/0.24	-1.72/0.06	7.41/0.24	2.28/0.04
	2 0	33.10/23.3	21.92/0.31	-0.49/0.14	11.13/0.31	2.06/0.03	12.70/0.30	0.59/0.02	4.11/0.31	-1.73/0.08	7.48/0.30	2.25/0.05
	2 5	33.91/23.8	21.92/0.38	-0.49/0.02	11.12/0.38	2.06/0.03	12.70/0.37	0.59/0.03	4.10/0.37	-1.73/0.09	7.53/0.36	2.21/0.06

Tab. 5: Results for the medium interference case. For each each M-estimator, the table reports the number of iterations required to converge, and the distribution (summarized by the average value and standard deviation) of the estimated coefficients for the power-line interference $A\,m(t)$.

Method	Noise	Iterations	50 Hz		100 Hz		150 Hz		200 Hz		250 Hz	
		#/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.
G.Truth	–	-/-	4.37 μV	-0.49 rad	2.23 μV	2.06 rad	2.54 μV	0.59 rad	0.82 μV	-1.74 rad	1.46 μV	2.27 rad
L1-L2	0 0	34/-	4.40/-	-0.49/-	2.22/-	2.06/-	2.52/-	0.59/-	0.86/-	-1.62/-	1.74/-	1.84/-
	0 1	23.73/ 7.16	4.40/0.01	-0.49/0.00	2.22/0.01	2.06/0.00	2.52/0.01	0.59/0.00	0.86/0.01	-1.63/0.02	1.75/0.02	1.82/0.01
	0 5	24.59/11.15	4.41/0.07	-0.49/0.02	2.22/0.07	2.06/0.03	2.52/0.07	0.59/0.03	0.86/0.07	-1.64/0.08	1.79/0.08	1.79/0.05
	1 0	23.91/ 9.88	4.41/0.14	-0.49/0.03	2.22/0.14	2.06/0.06	2.53/0.14	0.59/0.06	0.86/0.14	-1.65/0.17	1.86/0.14	1.74/0.08
	1 5	22.95/ 8.51	4.42/0.21	-0.49/0.05	2.22/0.21	2.06/0.09	2.53/0.21	0.58/0.08	0.87/0.20	-1.67/0.25	1.96/0.21	1.68/0.11
	2 0	22.10/ 7.06	4.42/0.27	-0.49/0.06	2.23/0.27	2.06/0.13	2.54/0.27	0.59/0.11	0.89/0.26	-1.68/0.36	2.08/0.28	1.62/0.14
	2 5	21.31/ 5.81	4.43/0.34	-0.49/0.08	2.24/0.34	2.06/0.15	2.55/0.33	0.58/0.14	0.91/0.32	-1.67/0.53	2.19/0.34	1.58/0.16
Huber	0 0	17/-	4.41/-	-0.49/-	2.22/-	2.06/-	2.52/-	0.59/-	0.89/-	-1.57/-	1.65/-	1.97/-
	0 1	23.65/11.07	4.41/0.01	-0.49/0.00	2.22/0.01	2.06/0.01	2.52/0.01	0.59/0.01	0.89/0.01	-1.57/0.02	1.66/0.02	1.96/0.02
	0 5	26.67/14.95	4.42/0.07	-0.49/0.02	2.22/0.07	2.06/0.03	2.53/0.07	0.59/0.03	0.89/0.07	-1.59/0.08	1.69/0.08	1.92/0.06
	1 0	26.04/14.28	4.42/0.14	-0.49/0.03	2.22/0.14	2.06/0.07	2.54/0.14	0.59/0.06	0.89/0.14	-1.63/0.17	1.74/0.15	1.86/0.10
	1 5	25.45/13.69	4.42/0.21	-0.49/0.05	2.23/0.21	2.06/0.10	2.54/0.21	0.59/0.08	0.89/0.21	-1.66/0.26	1.84/0.22	1.78/0.13
	2 0	24.52/12.38	4.42/0.28	-0.49/0.06	2.23/0.28	2.06/0.13	2.55/0.28	0.59/0.11	0.89/0.27	-1.68/0.37	1.96/0.28	1.71/0.15
	2 5	23.64/10.98	4.43/0.34	-0.49/0.08	2.24/0.34	2.06/0.16	2.55/0.34	0.58/0.14	0.91/0.32	-1.67/0.56	2.08/0.35	1.64/0.18
Tukey	0 0	22/-	4.34/-	-0.49/-	2.28/-	2.05/-	2.57/-	0.59/-	0.99/-	-1.57/-	1.48/-	2.59/-
	0 1	29.31/14.03	4.34/0.02	-0.49/0.00	2.28/0.02	2.05/0.01	2.57/0.02	0.59/0.01	0.99/0.02	-1.57/0.02	1.48/0.02	2.59/0.02
	0 5	37.81/24.27	4.35/0.08	-0.49/0.02	2.27/0.08	2.05/0.04	2.57/0.09	0.59/0.03	0.99/0.08	-1.58/0.09	1.50/0.09	2.57/0.07
	1 0	37.84/24.80	4.37/0.16	-0.49/0.04	2.26/0.16	2.05/0.08	2.56/0.17	0.59/0.06	0.96/0.16	-1.62/0.18	1.53/0.17	2.46/0.13
	1 5	36.10/24.12	4.41/0.24	-0.48/0.05	2.25/0.24	2.05/0.11	2.55/0.24	0.59/0.09	0.91/0.24	-1.66/0.29	1.59/0.24	2.33/0.18
	2 0	35.30/23.66	4.44/0.30	-0.47/0.07	2.25/0.30	2.05/0.14	2.56/0.30	0.59/0.12	0.88/0.30	-1.68/0.46	1.68/0.31	2.19/0.22
	2 5	36.47/24.46	4.44/0.38	-0.47/0.08	2.26/0.37	2.06/0.17	2.56/0.37	0.59/0.15	0.89/0.34	-1.64/0.70	1.78/0.37	2.03/0.28

Tab. 6: Results for the low interference case. For each each M-estimator, the table reports the number of iterations required to converge, and the distribution (summarized by the average value and standard deviation) of the estimated coefficients for the power-line interference $A\,m(t)$.

Method	Noise	Iterations	50 Hz		100 Hz		150 Hz		200 Hz		250 Hz	
		#/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.	α/std.dev.	φ/std.dev.
G.Truth	–	-/-	2.19 μV	-0.49 rad	1.11 μV	2.06 rad	1.27 μV	0.59 rad	0.41 μV	-1.73 rad	0.73 μV	2.27 rad
L1-L2	0 0	36/-	2.22/-	-0.49/-	1.11/-	2.06/-	1.25/-	0.58/-	0.45/-	-1.52/-	1.11/-	1.56/-
	0 1	24.51/ 7.53	2.22/0.01	-0.49/0.00	1.11/0.01	2.06/0.01	1.25/0.01	0.58/0.01	0.45/0.01	-1.53/0.03	1.13/0.02	1.55/0.02
	0 5	25.39/11.51	2.22/0.07	-0.49/0.03	1.11/0.07	2.06/0.06	1.25/0.07	0.58/0.06	0.45/0.07	-1.54/0.16	1.19/0.08	1.51/0.07
	1 0	24.71/10.55	2.23/0.14	-0.49/0.06	1.11/0.14	2.06/0.13	1.26/0.14	0.58/0.11	0.47/0.14	-1.57/0.34	1.29/0.15	1.45/0.12
	1 5	23.81/ 8.93	2.23/0.21	-0.49/0.09	1.12/0.20	2.06/0.19	1.27/0.21	0.58/0.17	0.49/0.19	-1.54/0.67	1.42/0.22	1.39/0.15
	2 0	22.80/ 7.35	2.24/0.27	-0.49/0.12	1.13/0.27	2.06/0.27	1.29/0.27	0.58/0.22	0.53/0.24	-1.44/1.00	1.56/0.28	1.34/0.18
	2 5	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.-	n.a.	n.a.	n.a.
Huber	0 0	17/-	2.22/-	-0.50/-	1.10/-	2.06/-	1.25/-	0.60/-	0.49/-	-1.43/-	0.97/-	1.76/-
	0 1	24.39/11.43	2.23/0.02	-0.49/0.01	1.10/0.01	2.06/0.01	1.26/0.01	0.59/0.01	0.49/0.01	-1.44/0.03	0.99/0.02	1.74/0.03
	0 5	27.53/15.19	2.23/0.07	-0.49/0.03	1.11/0.07	2.06/0.07	1.26/0.07	0.59/0.06	0.49/0.07	-1.47/0.15	1.03/0.08	1.69/0.09
	1 0	27.05/14.87	2.23/0.14	-0.50/0.06	1.12/0.14	2.06/0.13	1.28/0.14	0.58/0.11	0.49/0.14	-1.53/0.33	1.12/0.16	1.60/0.15
	1 5	26.16/13.90	2.23/0.21	-0.49/0.10	1.13/0.21	2.06/0.20	1.29/0.21	0.58/0.17	0.51/0.20	-1.55/0.65	1.25/0.23	1.50/0.19
	2 0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
	2 5	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
Tukey	0 0	23/-	2.15/-	-0.50/-	1.16/-	2.05/-	1.30/-	0.59/-	0.59/-	-1.45/-	0.82/-	2.88/-
	0 1	30.50/14.49	2.15/0.02	-0.50/0.01	1.16/0.02	2.05/0.02	1.30/0.02	0.59/0.01	0.59/0.02	-1.45/0.03	0.82/0.02	2.88/0.04
	0 5	38.84/24.28	2.16/0.08	-0.50/0.04	1.16/0.08	2.05/0.08	1.30/0.09	0.59/0.06	0.59/0.08	-1.47/0.16	0.83/0.09	2.79/0.51
	1 0	38.81/24.78	2.19/0.16	-0.50/0.08	1.15/0.16	2.05/0.15	1.30/0.17	0.59/0.13	0.56/0.16	-1.53/0.35	0.84/0.17	2.49/0.90
	1 5	37.03/24.18	2.23/0.24	-0.48/0.11	1.15/0.24	2.05/0.22	1.30/0.24	0.60/0.19	0.53/0.22	-1.52/0.75	0.89/0.24	2.27/0.79
	2 0	36.19/23.72	2.26/0.31	-0.46/0.14	1.16/0.30	2.05/0.31	1.31/0.30	0.59/0.25	0.54/0.26	-1.33/1.20	0.98/0.30	2.06/0.67
	2 5	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.

the signals $b(t)$ and $n(t)$. More in general we can summarize the following conclusions:

- All the estimators have a (usually small) bias, even in absence of noise, probably due to the low sampling rate
- The components of the highest frequencies (200, 250 Hz) are not well fitted, probably because they are more contaminated by the Gaussian noise that is mainly located in the high frequency band.
- L1-L2 and Huber has comparable performance in terms of accuracy and average iterations
- The iteration distribution for Huber is wider than L1-L2 one
- Tukey requires the highest number of iterations to converge and is less precise as noise increases

From all this considerations, L1-L2 is found to be the best among the 3 estimators to compute the power-line interference.

3.3.4 Real data

The method proposed so far requires that the fundamental frequency of the power-line interference is known. In real situations, the actual frequency can be different from the nominal one (e.g. 50 or 60 Hz), and an estimate of it can be necessary to obtain a reliable performance of interference removal. To this aim a frequency estimator can be exploited, such as the lattice-based algorithm of [4] since it features instantaneous estimation of the frequency, desirable performance and low complexity. The frequency estimator can update continuously sample by sample the value of the fundamental frequency and provide that value to the robust linear regression algorithm of Section 3.3.2.

To see the effect of a wrong estimation of the frequency compared to using a correct value, consider the example given by a data chunk of 1 second taken from channel C3 of Fig. 10. Although in this case the EEG content is expected to be neglectable or null, an effective removal of the power-line interference is a good practise even for not so reliable channels like C3 in Fig. 10. When the robust notch algorithm (with L1-L2) uses the nominal value of 50 Hz, it is clearly visible that the interference is not properly removed at the two sides of the chunk where a residual waveform is still visible (Fig. 23); if instead the value of 50.027 Hz found by the lattice algorithm is used, the interference removal is good (Figs. 24 and 25). Even if the difference between the nominal and estimated values of the frequency is so small, there is a noticeable residual in the bin of the spectrum corresponding to 50 Hz for the former case (Fig. 23) that disappears almost completely in the latter (Figs. 24 and 25).

As final example, the frequency estimator was applied to a slice of 1 second taken from MDA cohort (patient no.5, SMR experiment with the heavyweight configuration at 256 Hz); the estimated value of the frequency (49.97 Hz) was used by the robust notch algorithm (with L1-L2). The result is plotted in Fig. 26: looking at the filtered spectrum (red line in the bottom picture), the bin components at 50 and 100 Hz become close to 0 as proof of the good performance of the filter.

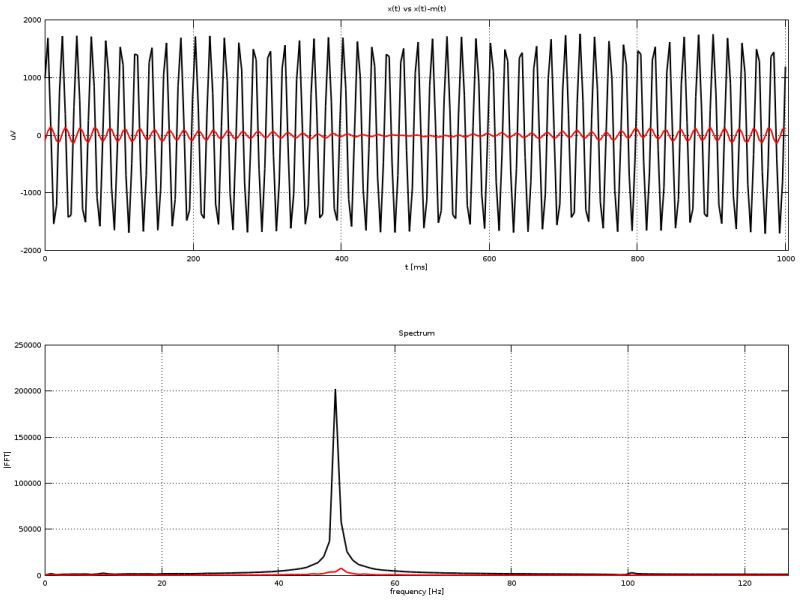


Fig. 23: The result obtained by L1-L2 estimator with the nominal value of 50 Hz for the fundamental power-line frequency; it is evident that, at the sides of the of the signal chunk, the interference is not removed. Top row: the signal $y(t)$ in black and the cleaned signal $y(t) - m(t)$ in red. Bottom row: corresponding Fourier spectrum for $y(t)$ and $y(t) - m(t)$.

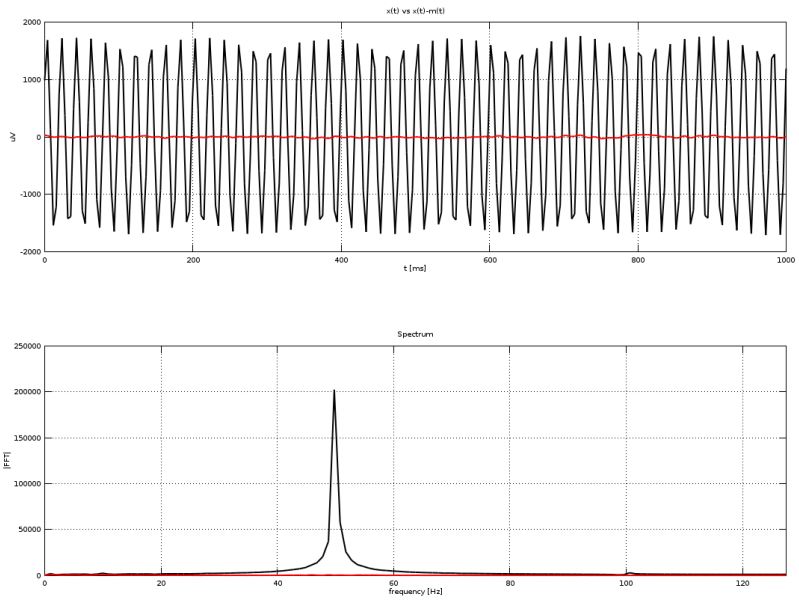


Fig. 24: The result obtained by L1-L2 estimator with the estimated value of 50.027 Hz for the fundamental power-line frequency; in this case the interference is properly removed. Top row: the signal $y(t)$ in black and the cleaned signal $y(t) - m(t)$ in red. Bottom row: corresponding Fourier spectrum for $y(t)$ and $y(t) - m(t)$.

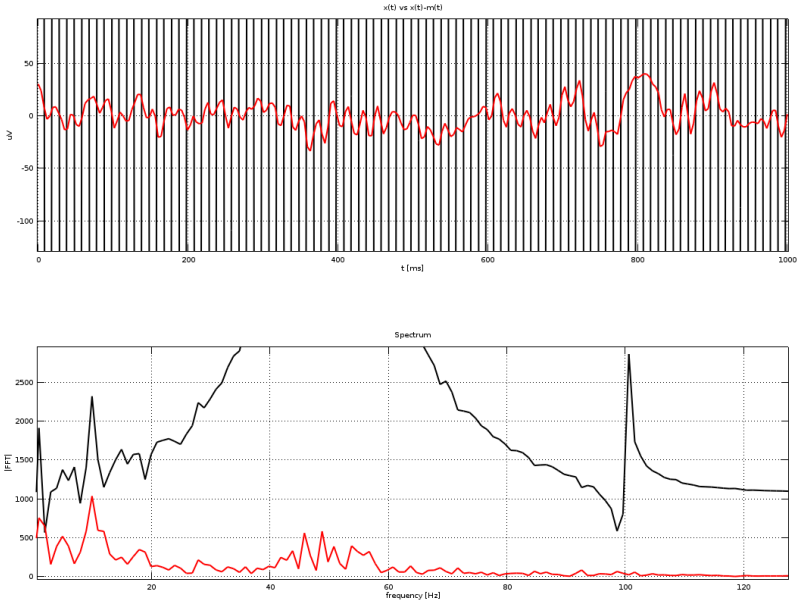


Fig. 25: A zoom in amplitude of Fig. 24.

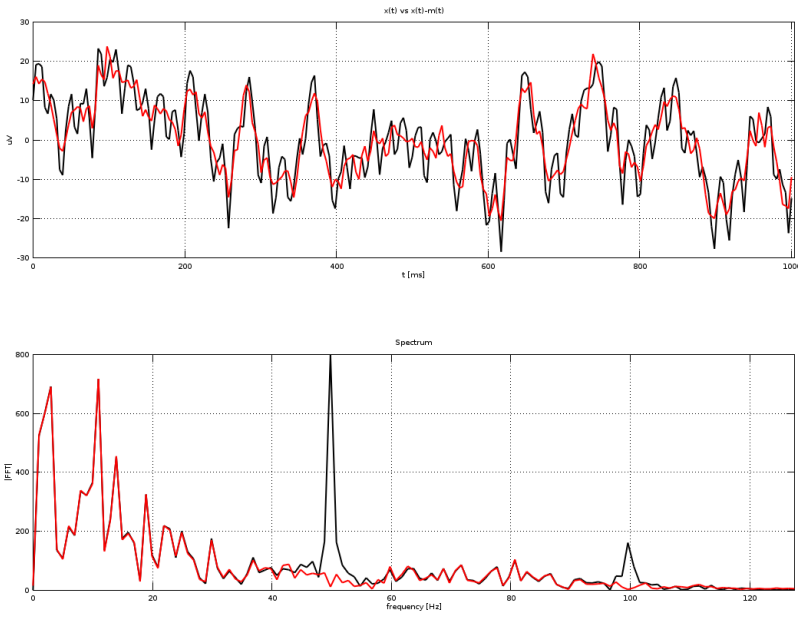


Fig. 26: The result obtained by L1-L2 estimator after computation of the fundamental power-line frequency by the lattice algorithm. Top row: the signal $y(t)$ in black and the cleaned signal $y(t) - m(t)$ in red. Bottom row: corresponding Fourier spectrum for $y(t)$ and $y(t) - m(t)$.

3.4 Final EEG filtering considerations for best practises

All the experiments carried out so far about EEG data preprocessing have highlighted the following conclusions:

- Wavelet-Amuse copes successfully with ocular artifacts, but not with most of other kind of noise;
- to avoid any risk to remove also the EEG signal, the suggestion is to fix only the first components (max 10 but less is better, e.g. 7)
- CAR filter should not be used if some channels are very noisy
- for Wavelet-Amuse, slices with length of 1-5 seconds are preferable (due to computational costs and effectiveness);
- for ErrPs detection, the use of Wavelet-Amuse does not worth, since in most of the cases it just slightly improves the accuracy of the detector;
- for artifacts different from the ocular ones, use ordinary high-pass and/or low-pass filters to remove the frequencies outside the band of interest;
- the robust notch algorithm presented here can remove the intense power-line interference better than standard notch, but it should be applied only to the (possibly few) channels of interest since it has a not so neglectable computational cost, and with data slices of about 1-2 seconds.

Successes and limits of Wavelet-Amuse This section is closed with an explanation why Wavelet-Amuse can remove the ocular artifacts but fails with other kind of noise. To this aim, Fig. 27 shows the first 14 components of a data chunk that contains both ocular and muscular artifacts at high frequencies (see Fig. 28). Usually the first components contain the ocular artifacts, and in this case they are clearly visible in the component no. 5 (but also in no. 2–4), while it is not clear where the high frequency noise goes. Hence the detection/correction of the ocular noise is easier than fixing the other kind of noise. To confirm this, Fig. 29 shows that the signals modified by the Wavelet-Amuse are without the ocular artifacts, but still contain the muscular ones.

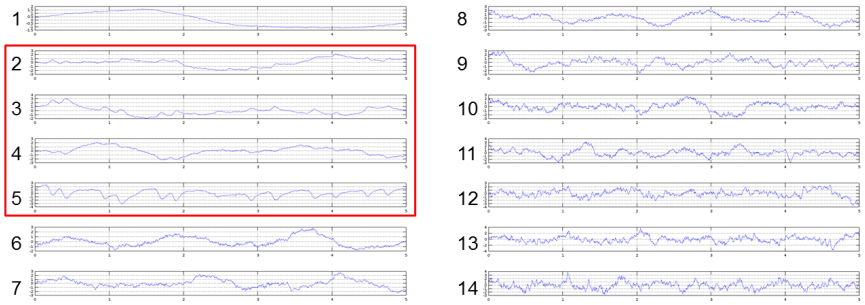


Fig. 27: First 14 independent components of an EEG data chunk with ocular and muscular artifacts (shown in Fig. 28). The red rectangle highlights the components where the ocular artifacts are mapped by Amuse; it is not clear which components are related with the muscular artifacts.

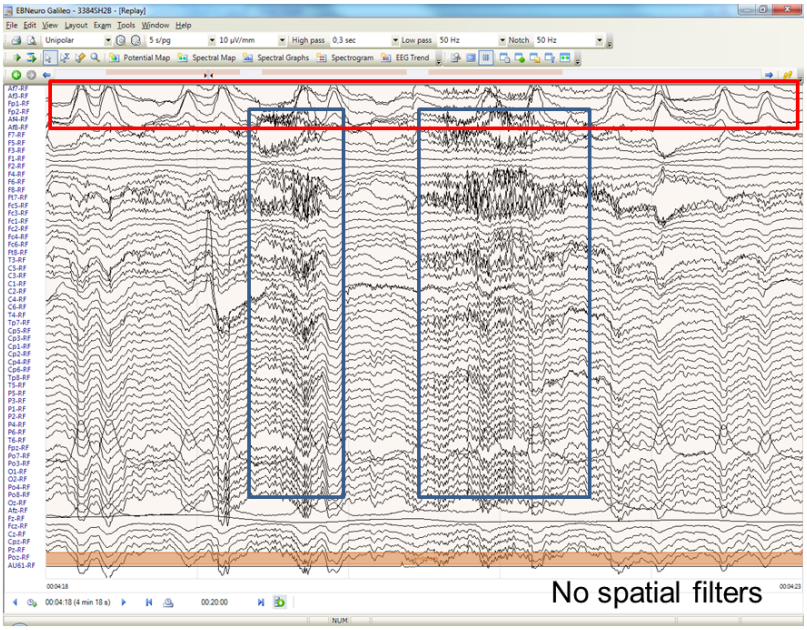


Fig. 28: An example of both ocular artifacts, pointed out by the red rectangle, and muscular artifacts, shown inside the two blue rectangles (from healty subject no. 2 of SHEBA cohort, SMR experiment with the heavyweight configuration).

Fig. 29: By fixing the first seven independent components, Wavelet-Amuse removes successfully the ocular artifacts (compare the channels inside the red rectangle in Fig. 28), while the muscular noise remains (blue rectangles).

4 EEG electrode monitoring

Aside trivial techniques such as signal thresholding, the aim of this section is to investigate an interesting and adaptable strategy to decide if an EEG data chunk is good or not for subsequent processing. Moreover, if the "bad data" condition is continuously active, it is likely that the channel has some problems to be fixed. Hence the method could substitute in a first run the role of the Ohmmeter check, i.e. the procedure to evaluate if the corresponding electrode to the channel is properly collecting the EEG signal.

The task of evaluating the quality of an EEG stream have been very challenging. To our knowledge, after a survey of the literature, only a recent paper focuses explicitly on this problem: [24], where six scores indicating the quality of the signal are proposed. The first score is calculated based on the general amplitude of the EEG channels. The second score is calculated based on which channel has the highest amplitude. The third score is based on calculating the dominant frequency for the channels. The last three scores depend on the amplitude and the geometrical shape of the main frequency bands (Delta, Theta, Alpha and Beta) of each channel. The paper shows that those scores are inversely related to the signal to noise ratio, but it does not provide any clear indication about the classification of abnormal behaviour of a channel.

Differently from [24], the strategy adopted here is based on a supervised classification approach. Furthermore, in order to be independent from the sampling rate, it was avoided to use descriptors made by segments of raw samples preferring descriptors belonging to the spectral domain.

4.1 A LDA approach

A preliminary study was based on the rational assumption that usually a "broken" channel collects a lot of power-line interference. Under this assumption, the power of the signal at 50 Hz P_p was compared to the average spectral power P_s without the contribution at 50 Hz.

Hence, a simple criterion could have been to put a threshold onto the ratio of P_p and P_s, $\rho = P_p/P_s$, but it had a quite poor performance.

For this reason, we moved to a more flexible methodology: after applying Fisher Linear Discriminant Analysis (LDA) [12] to a preparatory dataset containing labeled EEG chunks of good and bad examples (see Fig. 30), a linear criterion was found to decide if the content of the channel was good or not. LDA was performed on a logarithmic scale of the original values: $p_p = \log_{10} P_p$ and $p_s = \log_{10} P_s$, in order to get compact clusters.

Given a set of labeled 2D points $\mathbf{p}_n = [p_{s,n}\ p_{p,n}]^\top$, belonging to the set $\mathcal{D}_g$ (good chunks) or $\mathcal{D}_b$ (bad chunks), the aim of LDA was to find a vector $\mathbf{w} = [w_1\, w_2]^\top$ such that the corresponding $y_n = \mathbf{w}^\top \mathbf{p}_n \in \mathbb{R}$ are divided into two sets $\mathcal{Y}_g$ and $\mathcal{Y}_b$ that are maximally separable (Fig. 31). After that, the optimal decision threshold w_t between $\mathcal{Y}_g$ and $\mathcal{Y}_b$ was found at the intersection of the one-dimensional distributions of the two projected sets (see again Fig. 31).

4.1.1 Interpretation of the LDA criterion

An interpretation of the LDA criterion for this classification task is the following.

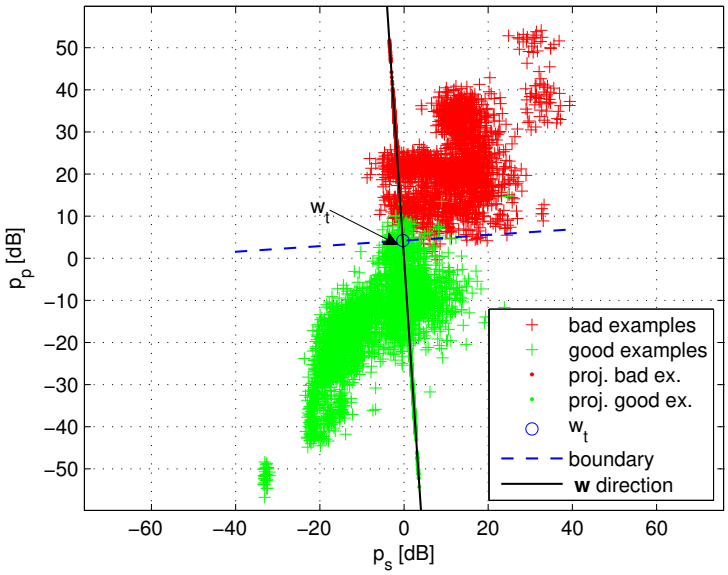


Fig. 30: LDA performed on labeled couples $(p_s,\, p_p)$ of bad and good EEG chunks $(\mathbf{w} = [0.067 \;\; -0.998]^\top)$. For visualization purposes, data are shifted into the centroid of the dataset by subtracting $\mu_c = [12.48\ 44.84]^\top$ dB.

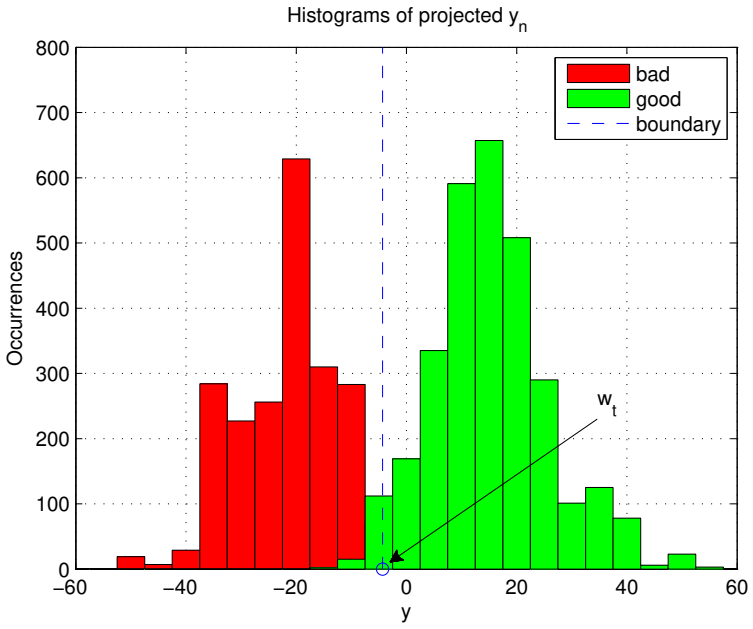


Fig. 31: Histograms of the two projected sets $\mathcal{Y}_b$ and $\mathcal{Y}_g$. By fitting on them two Gaussian distributions, the optimal decision threshold was found at $w_t = -4.32$.

Let's start with the equation of the optimal decision boundary:

$$\mathbf{w}^\top \mathbf{p} - w_t = 0$$

By expanding it we get,

$$w_1 p_s + w_2 p_p - w_t = 0 \Rightarrow p_p = -\frac{w_1}{w_2} p_s + \frac{w_t}{w_2} \tag{22}$$

that is the equation of the blue dotted line in Fig. 31, perpendicular to the direction of $\mathbf{w}$.

Now consider again the ratio ρ of P_p and P_s and take the logarithm of it:

$$\underbrace{\log_{10} \rho}_{\tau} = \log_{10} \frac{P_p}{P_s} = \log_{10} P_p - log_{10} P_s = p_p - p_s \Rightarrow p_p = ps + \tau \tag{23}$$

that represents a line with slope equal to 1 and offset τ. Thus, the simple criterion of fixing a threshold on the ratio of P_p and P_s is equivalent to move that line along the p_p axis.

By comparing Eq. 22 with Eq. 23, we can see that the LDA criterion adds one more degree of freedom to the rule based on the simple ratio, i.e. the slope of the line is not constrained anymore to be 1.

4.2 A more general solution

Although the method based on LDA was able to detect the bad EEG chunks due to power-line interference, the visual analysis of the exams contained in Phase I dataset showed that the power-line interference cannot be the only indicator of bad EEG data.

As a consequence, the analysis was expanded to the band 1-64 Hz (the whole available spectrum for a 128 Hz acquisition, the lowest sampling rate among the commonly used ones) and the Phase I dataset was used as source of bad and good examples. The more general assumption is that a "bad electrode" has a completely different spectrum compared to the spectrum of a good channel; furthermore, the generalization can cover both the case of 50 Hz and 60Hz for the power-line interference.

Due to the intensive use, at a certain point the heavyweight headcap used in the MDA and AUTH cohorts got a damage in C3 electrode (and sometimes other channels were not properly collecting the EEG signals). Thus, by manually labelling several (good and bad) channels from those acquisitions, both training and validation sets were created for a supervised classification approach. The same subject does not appear in both the datasets. Among the bad examples were inserted chunks containing strong power-line interference (Fig. 10), and highly disturbed data by external sources (e.g. Fig. 7) or intense muscular artifacts (Fig. 13); chunks with simple ocular artifacts (like in Fig. 4) instead were considered among the good examples, since it has been already shown that they can be handled by Wavelet-Amuse.

The training set contains 7563 examples while the validation set is made by 2456 examples; and each example contains 1 second of data coming from one of the selected channels. For each data chunk a descriptor with 63 components coming from FFT has been computed (the dc component was removed by subtracting the average value of the chunk).

Among the several supervised approaches, Support Vector Machines (SVMs) have been selected for they easiness of parameterization and use: a good performance both for the training set (96.30% of correct classifications) and validation set (96.45%) was found.

4.2.1 SVM parametrization and training

Among the several kernels proposed for the SVM model, in this work was selected the Radial Basis Function (RBF) Kernel, very popular in literature [1]:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{e}^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

where the *width* parameter $\gamma > 0$ controls the width of the Gaussian kernel. The other parameter of the SVM to be tuned is a *cost* parameter $c > 0$ that imposes a penalty on misclassified samples (i.e. a regularizer).

Thus, the parameter space of the RBF-SVM is made only by the couple (γ, c); with such a small dimension of the parameter space, a grid-search approach is the standard method to train the SVM. As suggested in [18], the use of exponentially growing sequences of c and γ is a practical method to identify good parameters; since from a preliminary random search we found that the accuracy in the training set improved as γ and c increased, here the tested parameters have been $\gamma = 2^i$, $i = 0, \ldots, 16$ and $c = 2^j$, $j = -1, \ldots, 8$.

Fig. 32 shows in graphical way the classification accuracy on the training set of the SVM in function of several values for (γ, c). Fig. 33 plots the classification accuracy of the trained SVM when applied to the validation set. As expected, when the SVM has an accuracy very close or equal to 100%, a overfitting occurs since the corresponding accuracy on the validation set becomes quite low (around 68%). The best compromise, minimizing the difference between the accuracy of the two sets, is achieved for $\gamma = 8$ and $c = 0.5$: as already written, the accuracy on the training set is 96.30%, while for the validation set it is 96.45%.

Tab. 7 shows the confusion matrix of the best SVM, both for the training and validation sets; in both cases, the errors are mostly due to false positives (246/7563=3.25% for the training set and 83/2456=3.38% for the validation set), but the error rate could be reduced by requiring that the "bad channel" condition should hold for $N > 1$ continuous times. The last two columns of the table shows that the classification score does not change significantly when the trained SVM at 256 Hz is called to evaluate the validation set sub-sampled at 128 Hz (95.32% vs. the the previous value of 96.45%), as a confirmation that the proposed strategy can cope with different sampling rates.

Tab. 7: Confusion matrix of the best SVM, for the training and validation sets at 256 Hz, and for the validation set at 128 Hz classified by the SVM trained at 256 Hz.

	Train. set (256 Hz)		Valid. set (256 Hz)		Valid. set (128 Hz)	
	good	bad	good	bad	good	bad
good	3392	246	924	83	892	115
bad	34	3891	4	1445	0	1449

After that, the trained SVM was run with some channels coming from other acquisitions of the SHEBA cohort; those acquisitions were never seen by the

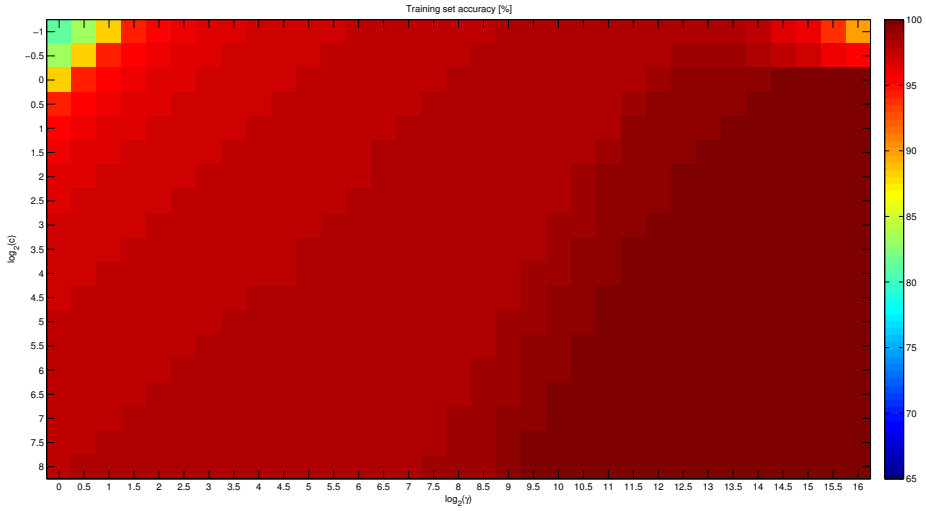


Fig. 32: Classification accuracy for the training set in function of various values for c and γ; ticks are labeled as $\log_2(\gamma)$ (x axis) and $\log_2(c)$ (y axis). Classification accuracy ranges from 81.145% to 100%. (best viewed in color)

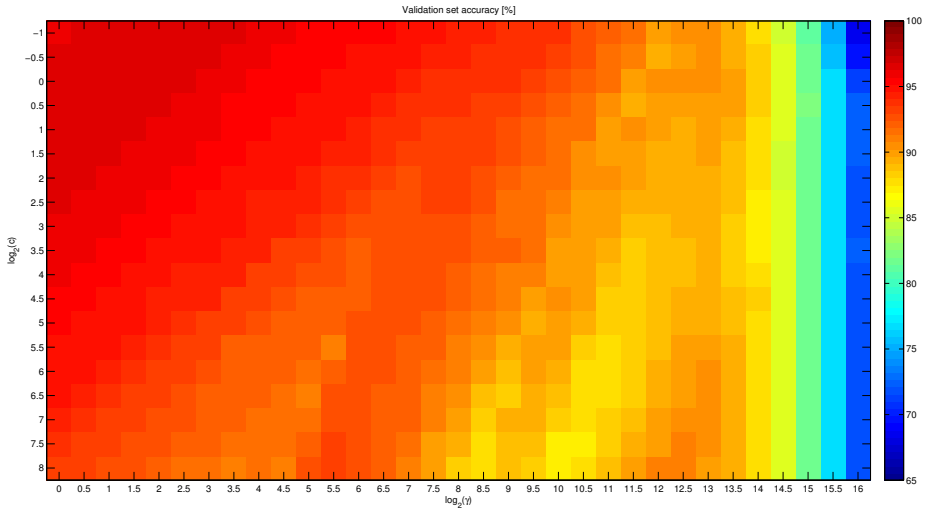


Fig. 33: Classification accuracy for the validation set in function of various values for c and γ; ticks are labeled as $\log_2(\gamma)$ (x axis) and $\log_2(c)$ (y axis). Classification accuracy ranges from 68.111% to 96.539%. (best viewed in color)

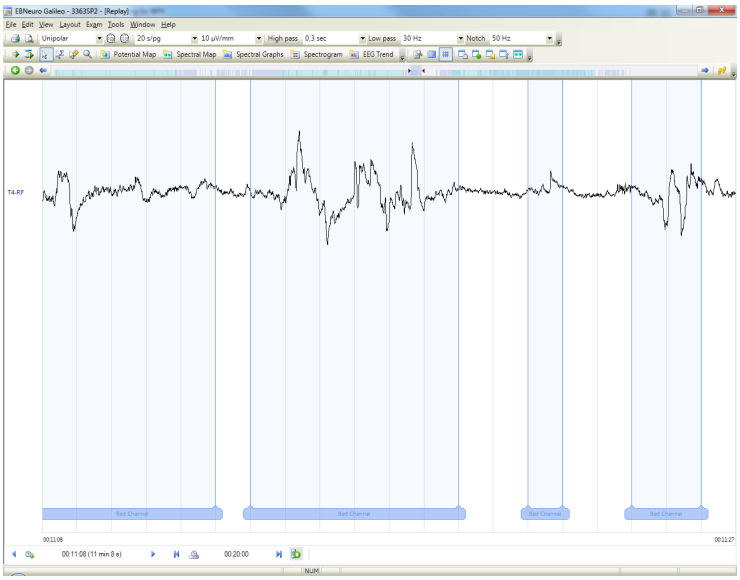


Fig. 34: Results for the classification of the channel quality of an exam from the Sheba cohort (patient no. 2, SMR experiment with heavyweight configuration), never used for training neither validation. Blue sections are labeled as bad data.

SVM, and the results were quite good as well. By visual inspection, there were of course some false positive/false negative but in general the classifier rose up the "bad data" label (blue sections in Fig. 34 as example) in points where the signal was not good.

All these experiments have shown that the SVM approach has good performance, that probably could be also improved by adding more examples to the dataset. A library realizing the described approach has been developed with a C interface; for the implementation details see appendix. B.

5 Final tuning about GSR data acquisition and best practices

An investigation of the data gathered during the Phase I trials was performed to assess the quality of the signals.

Initially, we investigated the signals visually. Based on this investigation, we observed that the signals exhibited unrealistic high-low peaks of conductivity levels which can be attributed to the loss of contact of the GSR sensors or by physical movements of the participants. This is evident if we consider that the typical skin resistance levels lie between 47kΩ and 1MΩ [2] and that in some cases the readings showed extreme levels of resistance which is the expected result of contact loss. Moreover, high frequency noise may be considered as evidence of motion artifacts that in most cases can be observed during the SMR phase of the experiment and for healthy subjects during which the participants were required to perform physical movement (fist clenching).

To remove these artifacts from the signals we applied the following artifact removal algorithm. To make our calculations easier we downsampled the recorded signals from 256Hz to 32Hz, which based on the literature [17] is more than enough for maintaining all the useful information of the GSR signal. Then we considered resistance measurements below 10kΩ as a recording error of the device (based on the device specifications as mentioned on the user manual [29]) and measurements above 1MΩ as the loss of contact of one or both sensors. Every value that was measured outside the range of 10kΩ–1MΩ was replaced with the mean of the previous 10 samples. An example of such contact-loss artifacts can be seen in Fig. 35.

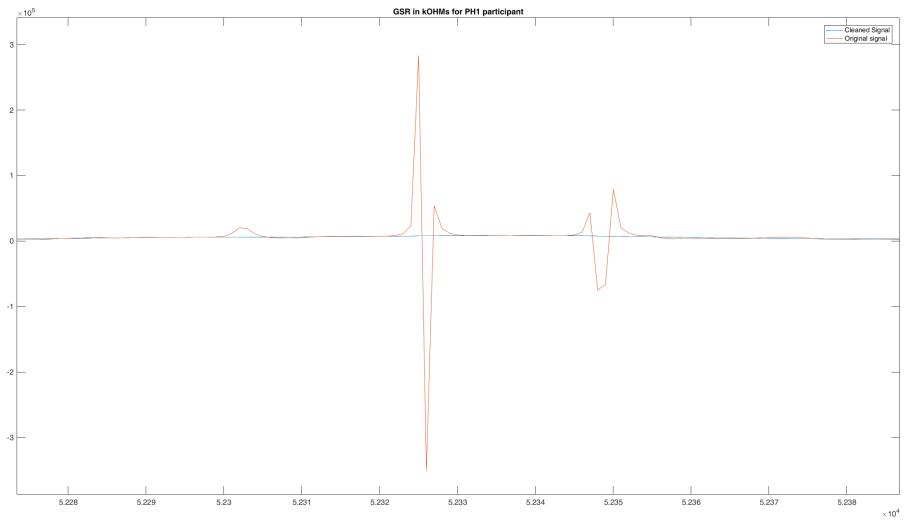


Fig. 35: Example of contact loss occurrences shown with the red line, and corrected signal (blue line). Note that the scale of y-axis is very high and so details of the GSR signal are not visible.

Moreover, in order to remove high frequency noise resulting from motion artefacts or other unknown sources, we applied a low-pass FIR filter with a

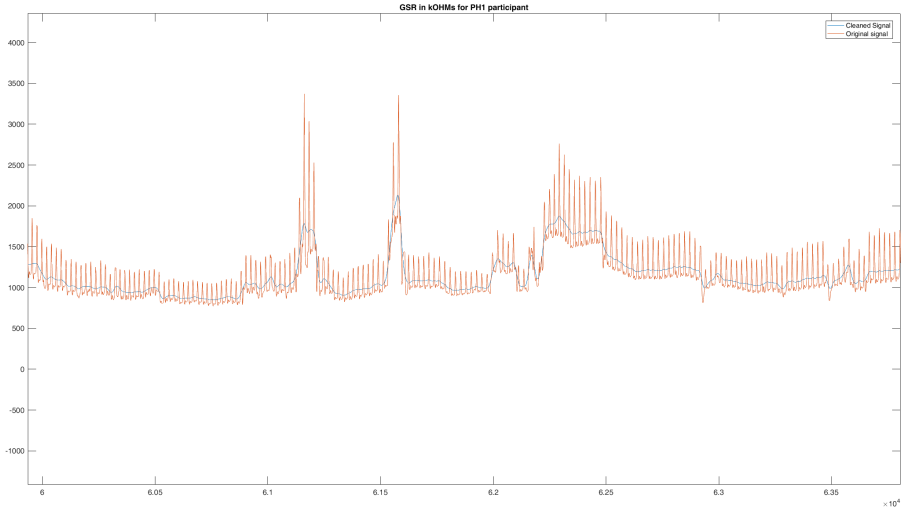


Fig. 36: Removing high frequency noise. Red line is the original signal and blue line the signal after low-pass filtering.

cut-off frequency of 5Hz using zero-phase filtering [23] so as to not introduce delays in our signals or distort the signals. An example of a signal with motion artifacts and after the proposed filtering approach can be seen in Fig. 36.

A Implementation of the robust notch filter

The EEG filter library has been enriched with the new functions for the robust notch filter algorithm of Section 3.3 (Math.NET Numerics [27] is used for internal matrix computations), together with the lattice-based frequency estimator[10] used in Section 3.3.4.

First of all, the settings for the robust notch filter and lattice-based frequency estimator are given by the structures `ROBUST_NOTCH_FILTER_SETTINGS` and `FREQUENCY_ESTIMATOR_SETTINGS` in List. 1. While the settings for the robust notch filter are self-explanatory, a description is required for some of the parameters of lattice-based frequency estimator: in particular `B0`, `Binf` and `Bst` manage the bandwidth of the frequency estimator, while `P0`, `Pinf` and `Pst` rule the settling time of the frequency estimation; usual values for such parameters are reported in Tab. 8 [20].

Parameter `B0` controls the initial notch bandwidth. Larger values of `B0` are preferred (e.g. tens of Hz) to achieve a faster initial convergence. Similarly, `Binf` controls the asymptotic notch bandwidth. Small values of `Binf` are preferred (e.g. tenths of Hz) to achieve more accurate estimation of the frequency. `Bst` controls the rate of transition between initial notch bandwidth `B0` and the asymptotic notch bandwidth `Binf` (i.e. the rate of convergence to 95% of the asymptotic bandwidth `Binf`).

Parameter `P0` controls the initial settling time of the frequency estimation algorithm; smaller values of `P0` are preferred (e.g. tenths of seconds) to achieve a faster initial convergence. Similarly, `Pinf` controls the asymptotic settling time of the frequency estimation algorithm. Considering the fact that the power-line frequency drifts are slow, larger values of `Pinf` are preferred (e.g. a few seconds) to obtain a more accurate estimation of the power-line frequency. `Pst` controls how fast the settling time changes from the initial value of P0 to its final value of `Pinf` (i.e. the rate of convergence to 95% of the asymptotic settling time `Pinf`); this transition time should be set large enough, e.g. a few seconds depending on the notch bandwidth, to allow global convergence.

Listing 1: Settings for the robust notch algorithm and lattice-based frequency estimator

```
// settings for the robust notch filter
typedef struct
{
   // Fundamental frequency of powerline interference (50 or 60 Hz)
   double frequency;

   // number of frequencies to be removed (fundamental plus the other
       harmonics)
   // e.g. if frequency=50 and nFrequencies=2, the frequencies to be
   // removed are 50 and 100
   int nFrequencies;

   // sampling rate in Hz (e.g. 128)
   double samplingRate;
```

[10] The implementation found in [19] was used as main sketch to write our code for the lattice-based frequency estimator.

```
   // number of samples of the chunk data that will be cleaned
   int sliceLength;

   // If not equal to 0, the algorithm is enabled to estimate
   // also the DC component of the chunk data that will be cleaned
   int estimatedDC;

} ROBUST_NOTCH_FILTER_SETTINGS;


typedef struct
{
    // fundamental frequency of the powwer-line signal
   double frequency_Hz;

    // sampling rate
   int fs;

    // Initial bandwidth of the frequency estimator
   double B0;

    // Asymptotic bandwidth of the frequency estimator
   double Binf;

    // Rate of convergence to 95\% of the asymptotic bandwidth Binf
   double Bst;

    // Initial settling time of the frequency estimator
   double P0;

    // Asymptotic settling time of the frequency estimator
   double Pinf;

    // Rate of convergence to 95\% of the asymptotic settling time
   double Pst;

} FREQUENCY_ESTIMATOR_SETTINGS;
```

Tab. 8: Recommended coefficients for the lattice-based frequency estimator.

Parameter	Unit	range
B0	Hz	10.00 – 50.00
Binf	Hz	0.01 – 0.10
Bst	s	0.50 – 10.00
P0	s	0.01 – 0.50
Pinf	s	1.00 – 5.00
Pst	s	1.00 – 10.00

Finally, in List. 2 the functions to create and use the robust notch filter and the frequency estimator are shown. Please note that, although the lattice-based frequency estimator is not a filter, for sake of convenience it is regarded as a PFILTER reference as well.

Listing 2: Functions to use the robust notch algorithm and the lattice-based frequency estimator

```
// function that creates the object implementing the
// lattice-based powerline frequency estimator
MAMEM_FILTER_API PFILTER CreateFrequencyEstimator(const float frequency);

// function that instantiates a robust notch object, to be used as
// a PFILTER reference, given the info in the input parameter
// of type ROBUST_NOTCH_FILTER_SETTINGS
MAMEM_FILTER_API PFILTER FilterCreateRobustNotch(const
    ROBUST_NOTCH_FILTER_SETTINGS* settings);

// function that removes the powerline interference
// input PFILTER: the reference to the robust notch filter previously
// created with FilterCreateRobustNotch(const
    ROBUST_NOTCH_FILTER_SETTINGS*)
// double freq: the estimated fundamental frequency of the powerline
    interference
// float* data: float array with the channel chunk of data to be cleaned
// int dataLength: length of float array data
MAMEM_FILTER_API errno_t RemovePowerLineByRobustNotch(PFILTER
    robustNotch, double freq, float* data, int dataLength);

// function that reset the lattice-based powerline frequency estimator
    represented by the reference PFILTER,
// previously created with CreateFrequencyEstimator(const float
    frequency);
MAMEM_FILTER_API errno_t FilterResetFrequencyEstimator(PFILTER
    frequencyEstimator);

// function that provides the value of the power-line frequency,
// countinuosly fed by data chunks in float* data
MAMEM_FILTER_API double FilterUpdateFrequencyEstimator(PFILTER
    frequencyEstimator, float* data, int dataLength);
```

B Implementation of the electrode monitor library

The EegMonitor library is a dll with a C interface: see List. 3 for a descriptions of the C functions and their parameters. The code behind the C interface exploits also the libSVM library [3], to include a class implementing the SVM functions, and Math.NET Numerics [27] for the FFT computation (used to create the descriptors for the SVM)

Listing 3: C interface for the EEG electrode monitoring functions

```
#ifdef MAMEM_EEG_MONITOR_API_EXPORT
#define MAMEM_EEG_MONITOR_API _declspec(dllexport)
#else
#define MAMEM_EEG_MONITOR_API _declspec(dllimport)
#endif

// ``pointer'' to underlying eeg monitor object
typedef void* PMONITOR;

extern "C"
{
   // This function creates the eeg monitor object.
   MAMEM_EEG_MONITOR_API PMONITOR EegMonitorCreate(double
       exampleDuration_ms, double samplingRate_Hz, double
       maxFrequencyInDescriptor_Hz);

   // This function destroys the eeg monitor object.
   MAMEM_EEG_MONITOR_API errno_t EegMonitorDispose(PMONITOR eegMonitor);

   // This function allows to adapt a trained channel monitor
   // to a sampling rate different from the one used in the
   // training session
   MAMEM_EEG_MONITOR_API errno_t UpdateSamplingRate(PMONITOR eegMonitor,
       double samplingRate);

   // The function returns in samplingRate the value of
   // the sampling rate of the channel monitor instance
   // given by PMONITOR
   MAMEM_EEG_MONITOR_API errno_t GetSamplingRate(PMONITOR eegMonitor,
       double* samplingRate);

   // function to train a channel monitor instance (eegMonitor)
   // given a training file in libSVM format
   // c and gamma represents the tranining parameters of the
   // RBF_SVM model adopted
   MAMEM_EEG_MONITOR_API errno_t Train(PMONITOR eegMonitor, char*
       datafile, double c, double gamma);

   // this function saves to file the state of a trained
   // channel monitor instance
   MAMEM_EEG_MONITOR_API errno_t SaveTrainedMonitor(PMONITOR eegMonitor,
       char* datafile);
```

```
    // The function Load a trained channel monitor from file
  static public ChannelMonitor LoadTrainedMonitor(string filename)

  // the function returns true in boolean entry isok if
  // the data length of the array to be analyzed agrees
  // with the lenght set for the PMONITOR instance of
  // channel monitor
  MAMEM_EEG_MONITOR_API errno_t CheckDataLength(PMONITOR eegMonitor,
      int length, bool& isok);

  // Function that analizes the float array of data and
  // provides in boolean entry isgood the value true is the
  // signal is good for EEG processing
  MAMEM_EEG_MONITOR_API errno_t EvaluateGoodChannel(PMONITOR
      eegMonitor, bool* isgood, float* data, size_t dataLength);

  // Function that return a string containing the descriptor
  // extracted for a EEG chunk of data. Useful to create new
  // training files
  MAMEM_EEG_MONITOR_API errno_t TextRawDescriptor(PMONITOR eegMonitor,
      float* data, size_t dataLength, bool goodChannel, char* text,
      int* charLength);
}
```

References

[1] M. Boardman and T.P. Trappenberg. A heuristic for free parameter optimization with support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006*, 2006.

[2] J.T. Cacioppo, L.G. Tassinary, and G. Berntson. *Handbook of Psychophysiology*. Cambridge University Press, 2007.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] N.I. Cho and S.U. Lee. On the adaptive lattice notch filter for the detection of sinusoids. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(7):405–416, 1993.

[5] MAMEM Consortium. D2.1: Prototype modules implementation for signal capturing. mklab.iti.gr/mamem/images/f/fd/D2.1_Sensor_configuration_and_signal_capturing_final.pdf.

[6] MAMEM Consortium. D2.2: Initial integration and optimization of multimodal sensors. mklab.iti.gr/mamem/images/2/23/D2.2_Initial_%CE%99ntegration_Optimization_Multi-modal_Sensors_Final.pdf.

[7] MAMEM Consortium. D3.1: Eye-tracking-, EEG- and biofeedback-based control with meso level control paradigms. mklab.iti.gr/mamem/images/b/bb/D3.1_BCI_Interaction_final.pdf.

[8] MAMEM Consortium. D4.2: Concepts and guidelines for technical verification and system integration. mklab.iti.gr/mamem/images/b/bf/D4.2_AIV_Plan_final.pdf.

[9] MAMEM Consortium. MAMEM EEG SSVEP Dataset I (256 channels, 11 subjects, 5 frequencies presented in isolation). https://figshare.com/articles/MAMEM_EEG_SSVEP_Dataset_I_256_channels_11_subjects_5_frequencies_/2068677, 2016.

[10] MAMEM Consortium. MAMEM EEG SSVEP Dataset II (256 channels, 11 subjects, 5 frequencies presented simultaneously). https://figshare.com/articles/MAMEM_EEG_SSVEP_Dataset_II_256_channels_11_subjects_5_frequencies_presented_simultaneously_/3153409, 2016.

[11] MAMEM Consortium. MAMEM Phase I Dataset - A dataset for multimodal human-computer interaction using biosignals and eye tracking information. https://figshare.com/articles/MAMEM_Phase_I_Dataset_-_A_dataset_for_multimodal_human-computer_interaction_using_biosignals_and_eye_tracking_information/5231053, 2017.

[12] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[13] EB Neuro SpA. Galileo software. www.ebneuro.biz/en/neurology/ebneuro/galileo-suite.

[14] A.M. Feit, S. Williams, A. Toledo, A. Paradiso, H. Kulkarni, S. Kane, and M.R. Morris. Toward everyday gaze input. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI'17*, 2017.

[15] J. Fox and S. Weisberg. Robust Regression in R – a web appendix to An R Companion to Applied Regression, second edition. https://socialsciences.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Robust-Regression.pdf, 2010.

[16] A.M. Gad and M.E. Qura. Regression estimation in the presence of outliers: A comparative study. *International Journal of Probability and Statistics*, 5(3):65–72, 2016.

[17] L.A. Geddes and L.E. Baker. *Principles of applied biomedical instrumentation*. Wiley, 1975.

[18] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, University of National Taiwan, East Lansing, Michigan, May 2016. www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[19] M.R. Keshtkaran and Z. Yang. Adaptive powerline interference canceller source code. https://github.com/mrezak/removePLI.

[20] M.R. Keshtkaran and Z. Yang. A fast, robust algorithm for power line interference cancellation in neural recording. *Journal of neural engineering*, 11(2), 2014.

[21] C. Levkov, G. Mihov, R. Ivanov, I. Daskalov, I. Christov, and I. Dotsinsky. Removal of power-line interference from the ECG: a review of the subtraction procedure. *BioMedical Engineering OnLine*, 4(1), 2005.

[22] E. Malis and E. Marchand. Experiments with robust estimation techniques in real-time robot vision. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'06*, 2006.

[23] The Mathworks. www.mathworks.com/help/signal/ref/filtfilt.html.

[24] S. Mohamed, S. Haggag, S. Nahavandi, and O. Haggag. Towards automated quality assessment measure for EEG signals. *Neurocomputing*, 237(Supplement C):281 – 290, 2017.

[25] Neuroelectrics. www.neuroelectrics.com/products/enobio/enobio-8/.

[26] S. Nikolopoulos, P.C. Petrantonakis, K. Georgiadis, F. Kalaganis, G. Liaros, I. Lazarou, K. Adam, A. Papazoglou-Chalikias, E. Chatzilari, V.P. Oikonomou, C. Kumar, R. Menges, S. Staab, D. Müller, K. Sengupta, S. Bostantjopoulou, Z. Katsarou, G. Zeilig, M. Plotnik, A. Gotlieb, R. Kizoni, S. Fountoukidou, J. Ham, D. Athanasiou, A. Mariakaki, D. Comanducci, E. Sabatini, W. Nistico, M. Plank, and I. Kompatsiaris. A multimodal dataset for authoring and editing multimedia content: The MAMEM project. *Data in Brief*, 15(Supplement C):1048 – 1056, 2017.

[27] Math.NET Numerics. https://numerics.mathdotnet.com/.

[28] R.M. Rangayyan. *Biomedical Signal Analysis*. Wiley, second edition, 2015.

[29] Shimmer. www.shimmersensing.com/images/uploads/docs/GSR_User_Guide_rev1.12.pdf.

[30] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76, 1997.