Technische Universität Berlin

Faculty IV - Electrical Engineering and Computer Science

Research Article

Machine learning for neurophysiological assessment of proximity in a virtual environment

Emile Gaudinot[1]
Sebastian Bosse[1]
Birgit Nierula[1]
Prof. Dr. Wojciech Samek[12]
Prof. Dr. Klaus-Robert Müller[12]

[1] Fraunhofer [2] Berlin Institute for the Foundations of Learning and Data

Submission
18.10.2025

Review
03.11.2025

Publication
10.12.2025

Declaration of Authorship

I hereby declare that the thesis submitted is my own, unaided work, completed without any external help. Only the sources and resources listed were used. All passages taken from the sources and aids used, either unchanged or paraphrased, have been marked as such. Direct quotations can be identified as follows: *"direct quotation"*.

Where generative AI tools were used, I have indicated the product name, manufacturer, the software version used, as well as the respective purpose: OpenAI - *ChatGPT* (4o, 4.1 & 5) and Mistral AI - *le Chat*. They were exclusively used for language and phrasing improvements. I am fully responsible for the selection, adoption, and all results of the AI-generated output I use.

I have taken note of the Principles for Ensuring Good Research Practice at TU Berlin dated 15 February 2023.

I further declare that I have not submitted the publication in the same or similar form to any other examination authority.

Berlin, 18.10.2025

Emile Gaudinot

Zusammenfassung

Diese Arbeit befasst sich mit dem dringenden Bedarf an einer automatisierten, modularen und echtzeitfähigen Pipeline zur Dekodierung von Elektroenzephalographie (EEG)-Signalen für die Nähebewertung. Obwohl zahlreiche Algorithmen zur EEG-Verarbeitung existieren, ist deren manuelle Verkettung zeitaufwendig und anfällig für Inkonsistenzen, was die Reproduzierbarkeit und Effizienz beeinträchtigt. Die zentrale Herausforderung bestand darin, eine robuste Toolbox zu entwickeln, die bestehende Methoden—primär aus `MNE-Python`—integriert und fehlende Funktionalitäten implementiert, um eine halbautomatisierte Pipeline für die EEG-Dekodierung zu schaffen. Die Lösung umfasste die Zusammenführung der Vorverarbeitungs-, Merkmalsextraktions- und Regressions-/Klassifikationsschritte in einem einheitlichen Framework mit Fokus auf Echtzeitanwendbarkeit und Modularität.

Zu den wichtigsten Erkenntnissen gehört die erfolgreiche Implementierung einer Pipeline, die die Vorverarbeitung von EEG-Signalen automatisiert, Spatio-Spectral Decomposition und Source Power Comodulation für die Merkmalsextraktion nutzt und Permutationsanalysen sowie Alignment-Algorithmen einsetzt, um signifikante Komponenten auszuwählen. Die Toolbox wurde anhand einer Fallstudie validiert, bei der EEG-Daten von Teilnehmenden verwendet wurden, die mit einem Avatar in einer virtuellen Realität interagierten, und demonstrierte dabei ihre Fähigkeit, Nähe zu dekodieren. Die Ergebnisse unterstreichen das Potenzial der Toolbox für breitere Anwendungen in der EEG-Analyse, Brain-Computer-Schnittstellen und der Echtzeitverarbeitung neurophysiologischer Signale, zeigen aber auch Bereiche für zukünftige Verbesserungen auf, wie die Erweiterung der Pipeline für die Integration multimodaler Daten.

Abstract

This thesis addresses the critical need for an automated, modular, and real-time-ready pipeline to decode electroencephalography (EEG) signals for proximity assessment. While numerous EEG processing algorithms exist, their manual chaining is time-consuming and prone to inconsistencies, hindering reproducibility and efficiency. The core challenge was to develop a robust toolbox that integrates existing methods—primarily from `MNE-Python`—and implements missing functionalities to create a semi-automated pipeline for EEG decoding. The solution involved wrapping the preprocessing, feature extraction, and regression/classification steps into a unified framework, with a focus on real-time applicability and modularity.

Key findings include the successful implementation of a pipeline that automates EEG signal preprocessing, leverages Spatio-Spectral Decomposition (SSD) and Source Power Comodulation (SPoC) for feature extraction, and employs permutation analysis and alignment algorithms to select significant components. The toolbox was validated using a case study involving EEG data from participants interacting with an avatar in a virtual reality environment, demonstrating its ability to decode proximity. The results highlight the toolbox's potential for broader applications in EEG analysis, brain-computer interfaces, and real-time neurophysiological signal processing, while also identifying areas for future refinement, such as extending the pipeline for multimodal data integration.

Acknowledgments

I would like to express my deepest gratitude to all those who have supported and guided me throughout the completion of this thesis.

First and foremost, I am sincerely grateful to **Birgit Nierula** for her invaluable time, regular meetings to frame the project, and her insightful inputs on proxemics and neuroscience. Her advice and precious feedback have been instrumental in improving the quality of this thesis.

I am also deeply indebted to **Sebastian Bosse** for offering me the opportunity to undertake this thesis, hosting me at Fraunhofer HHI, and helping me understand concretely abstract concepts and algorithms. His guidance and feedback have been crucial to the development of this work.

I extend my heartfelt thanks to **Wojciech Samek** and **Klaus-Robert Müller** for accepting to review my thesis. I am especially grateful to Klaus-Robert Müller for his courses *Machine Learning I* and *II* at the TU Berlin, which laid the foundations for my work here.

A special thank you to **Anna Melnik** for her assistance in understanding the SSD and SPoC algorithms, and to **Mert Akgül** for his precious help regarding any question.

Lastly, I would like to thank **Aileen, Alex, Dilara, Dorian, Farelle, George, Nikita, Phil, and Yichen** for the great time in the group and for making me feel so welcome.

This thesis would not have been possible without the support and encouragement of all these individuals.

Contents

While **Chapters 1, 2, 5, and 6** provide the context and discuss the work, **Chapters 3 and 4** are the core of this thesis.

List of Figures

List of Tables

1. Introduction

A data processing pipeline that relates neural oscillations to behavioral parameters does not exist yet. It is however essential and would speed up electroencephalography (EEG) analysis in many applications. This thesis is part of the K3VR project [24], which aims to develop a VR system to train police officers in handling conflict and escalation situations with the public through communication. Identifying escalation situations can be done through the detection of personal space (PS) intrusion—PS being the invisible boundary individuals maintain around themselves to feel comfortable and secure. This is the case study of this Master's thesis. This detection relies on the analysis of EEG data, which involves many processing steps. Although this processing is standard, and most of the functions already exist, no complete and automated pipeline exists yet. The processing steps have to be chained together manually, which is very time-consuming. This work focuses on the development of a toolbox that wraps existing methods and implements missing ones to create a semi-automated, modular, and real-time-ready pipeline for EEG decoding. The toolbox is designed to be generalizable, so it can be adapted for many applications involving EEG analysis.

Motivation Many EEG processing and decoding algorithms already exist, and are even implemented. However, not all of them are available, and to our knowledge, no wrapper implements the whole pipeline in Python, from A to Z. Even `MNE-Python`—the most common open-source library for EEG processing—implements each function separately, but does not provide a complete pipeline, and doesn't fit all the requirements of our case study. It is the same problem with `Wyrm` [56]. Researchers then have to manually chain all functions together—a repetitive and time-consuming process. Manual pipelines also struggle with reproducibility. From one study to another, the same systematic analysis may be implemented with different tools, different libraries, different methods, or in a different order, leading to different results.

Research Challenge Our challenge is thus to find the best combination of existing methods, and implement the missing ones, to create a robust and efficient toolbox which extends `MNE-Python` functions and machine learning algorithms. The final goal is to unify them into a complete pipeline that can be used with minimal user input, to decode EEG signals. The case study is the decoding of neurophysiological signals associated with perceived distance from noisy, high-dimensional EEG data, collected on subjects in a virtual environment.

Novelty The main contribution of this toolbox is to provide a complete pipeline from raw EEG data to decoded neurophysiological signals related to proximity and distance assessment. It can be easily extended to other problems. It includes machine learning algorithms and extends `MNE-Python`. The toolbox is almost fully automated and only relies on a few parameters to be defined at the beginning. It is also modular, so that only parts of the pipeline can be used if needed, without breaking it. In addition, this toolbox extends existing frameworks by integrating optimization methods, like the Hungarian algorithm. Finally, a framework to use this toolbox in real-time is implemented, which is the final goal for many applications, such as the K3VR project, where the VR scenario should adapt in real-time to the user's arousal state.

Anticipated Impact The outcomes of this research have implications for the K3VR project, as this toolbox serves as the basis for the analysis and use of EEG data to assess proximity. More generally, this toolbox can be used or adapted for other applications involving EEG analysis, both offline and in real-time, and for BCIs. The toolbox can be easily extended to decode other neurophysiological signals from different problems by changing only a few parameters at the beginning.

Structure of the Master's thesis This thesis is structured into six chapters:

- **Chapter 2 - Related Work:** Reviews existing literature and foundational concepts about existing toolboxes and pipelines.
- **Chapter 3 - The Toolbox:** Describes the toolbox itself, detailing the software environment, the structure of the toolbox, and explaining each functionality. It highlights how the pipeline can be extended.
- **Chapter 4 - Validation:** Presents the validation methodology—the case study—, the experimental setup, and discusses the results obtained from applying the toolbox to real EEG data.
- **Chapter 5 - Discussion:** Discusses the limitations of the approach and outlines directions for future research. It addresses challenges encountered during the study and suggests improvements.
- **Chapter 6 - Conclusion:** Concludes the thesis, summarizing key findings, implications, and considers the broader impact and ethical aspects of the work.

While **Chapters 1, 2, 5, and 6** provide the context and discuss the work, **Chapters 3 and 4** are the core of this thesis.

2. Related Work

2.1. The choice of Python

Python's widespread adoption by the public and researchers has solidified its position as a leading programming language, with a strong and active community, far outpacing proprietary alternatives like MATLAB. Written in Python, the toolbox can then easily be integrated in already existing software environments. In addition, Python is entirely free and open-source. This is a strong argument in favor of adopting this programming language: most people will then be able to use the toolbox, from industry teams to researchers and students. All of this makes Python not only a practical choice but also a sustainable one for long-term projects and collaborative research.

2.2. Existing EEG toolboxes

A toolbox is a set of reusable functions, scripts, or modules that simplify common or repetitive tasks. It avoids rewriting the same code by storing frequently used functions and speeds up development a lot. One very popular toolbox in Python is scikit-learn [40]. Here is an overview of existing EEG toolboxes in Python.

MNE-Python is one of the most widely used open-source Python packages for processing and analyzing EEG and MEG data [17]. It provides a comprehensive suite of tools for data preprocessing, visualization, source localization, and statistical analysis. MNE-Python is highly regarded for its robustness and extensive documentation, making it suitable for both beginners and advanced users. While it does not provide a complete, automated pipeline from raw data to decoded signals, it serves as a foundational library for many EEG analysis workflows.

Braindecode is a Python toolbox specifically designed for decoding raw electrophysiological brain data (EEG, ECoG, MEG) using deep learning models [51]. BioPyC is another Python toolbox for offline classification of EEG and physiological signals [2]. It is designed to support brain-computer interface (BCI) research. EEG-pyline is a toolbox for EEG data preprocessing, analysis, and visualization, created for neuroscience and mental health research [1]. It aims to make EEG analysis in Python easier for researchers who are not familiar with programming. All of these toolboxes however have the same problem as MNE-Python: they don't provide a ready-to-use workflow from raw data to decoded signals, which is what we need.

Finally, Wyrm is another open-source Python toolbox developed for the analysis of EEG data [56]. It is part of a broader Python ecosystem for open-source BCI

systems, together with Mushu [57] and Pyff [58]. It implements feature extraction, classification, and visualization of neural data. It supports both offline data analysis and real-time online experiments, making it ready for BCI applications. Thanks to its extensive use of unit testing, `Wyrm` ensures software reliability and minimizes defects. This toolbox would be a very good candidate for our problem. However, within the Interactive and Cognitive Systems group, a more extendable pipeline is necessary. There are in particular two requirements. It has to capture the future needs of incorporating other data modalities, and it has to deal with VR. These are the reasons why `Wyrm` doesn't perfectly fit our needs.

2.3. Existing pipelines

Pipelines are used to streamline and automate workflows, often wrapping functions of a toolbox. They chain together multiple steps (e.g., data preprocessing, feature extraction, decoding) into a single, cohesive process. As explained before, even if many toolboxes exist, this chaining process always has to be re-implemented. It can be very long. Here is a review of existing pipelines, which automate whole workflows in only a few commands and functions.

2.3.1. Pipelines for general tasks

A first example is an NLP pipeline with HuggingFace, on sentiment analysis. Analyzing feelings in a sentence involves a whole bunch of functions. However, automated in a single pipeline, it makes this task very fast, user-friendly and accessible to non-specialists, as shown in Code Fragment 1.

```python
from transformers import pipeline

# Load a predefined sentiment analysis pipeline
sentiment_pipeline = pipeline("sentiment-analysis")

# Use the pipeline
result = sentiment_pipeline("I love using predefined pipelines!")
print(result)  # Output: [{'label': 'POSITIVE', 'score': 0.9998}]
```

Code Fragment 1: Hugging Face Transformers: an NLP Pipeline.

2.3.2. Pipelines for EEG-specific tasks

Here is an overview of A to Z pipelines, for EEG tasks. Many of them are however implemented in `MATLAB`.

`pySPACE` enables users to build, configure, and execute complex signal processing and classification pipelines [26]. It however relies on YAML configuration files, which is not as user-friendly as a `Python` environment. In addition, it doesn't provide advanced visualization tools like interactive 3D brain plots, which are essential for detailed data exploration. `pySPACE` is more directed towards performance metrics plotting.

The Maryland Analysis of Developmental EEG `MADE` Pipeline [11] is designed for preprocessing and analyzing EEG data, specifically tailored for developmental studies (e.g., infants and children). It aims to address the unique challenges of working with EEG data from young populations, such as higher levels of noise and artifacts. We are however more interested in adult EEG processing. Our case study only relies on adults.

`NeuroKit2` is a user-centered `Python` package for neurophysiological signal processing, including EEG [33]. It offers high-level functions and validated pipelines for data processing, making it accessible for users with varying levels of programming experience. These pipelines—like `ecg_process()`, `emg_process()` or `eog_process()`—automate the whole workflow, including data cleaning, preprocessing and analysis. It is however not implemented for EEG data.

`NeuroPycon` is an open-source multi-modal brain data analysis toolkit that provides `Python`-based template pipelines for advanced multi-processing of MEG and EEG data, among others [9]. The `ephypype` package includes pipelines for electrophysiology analysis. It's based mainly on the `MNE-Python` package, as well as more standard `Python` libraries such as Numpy and Scipy. It is exactly what we are interested in. However, the ready-to-use pipelines it provides are not exactly what we want. The *preprocessing pipeline* runs the ICA algorithm for an automatic removal of eye and heart-related artifacts. We would prefer a semi-automatic approach for this crucial step—`NeuroPycon` doesn't allow for manual approval of the removed artifacts. The other pipelines don't relate to our problem. `NeuroPycon` unfortunately lacks flexibility.

The `FieldTrip-SimBio` pipeline integrates the FieldTrip toolbox with SimBio, a software for realistic volume conductor modeling [59]. `DISCOVER-EEG` is an automated pipeline specifically designed for clinical neuroscience [15]. It focuses on identifying EEG biomarkers—measurable indicators of neurological or psychiatric conditions, from EEG in BIDS format. `PREP` (Preprocessing Pipeline) is designed to standardize and automate the preprocessing of large EEG datasets [4]. It includes filtering, artifact detection and re-referencing. It can be adapted to different EEG systems and experimental designs. The last three pipelines are very interesting, but are unfortunately implemented in `MATLAB`.

3. The Toolbox

This section introduces the computational toolbox developed as part of this Master's thesis. Following a description of the software environment, an overview of the toolbox architecture is provided. Each component of the toolbox is then examined in detail, and the potential for pipeline extensibility is discussed. An example of the whole pipeline running on real EEG data is described in **Chapter 4 - Validation**.

3.1. Software environment

The software environment for the toolbox is based on `Python 3.13`. The main libraries used include `MNE` (version 1.10.1) and `MNE-LSL` (version 1.10.0) for EEG data processing and the real-time framework, `numpy` (version 2.0.2) for numerical computations and data structures, `scikit-learn` (version 1.6.1) for machine learning tasks, `scipy` (version 1.16.0) for scientific computing, `matplotlib` (version 3.10.1) for data visualization, and `pickle` for object serialization. All libraries are used in their respective stable versions to ensure reliability and reproducibility.

The toolbox builds on `MNE-Python`, and extends it to meet the requirements, using functions from all the libraries mentioned above.

3.2. Overview

Research problem The goal of this toolbox is to wrap existing methods (essentially from `MNE-Python`, but not only) and implement missing ones to create a robust and efficient pipeline for EEG decoding. The toolbox should be almost fully automated, modular, and ready for real-time use. It has to be ready to use the decoded EEG data for one particular application: assessing perceived proximity in a virtual environment.

Challenges and subproblems The first challenge is to wrap the preprocessing functions of `MNE-Python` to have a cleaned signal ready for feature extraction. The second challenge is to use the SSD and SPoC functions of `MNE-Python` in an automatic way, to be used for other tasks as well as our specific one. The third challenge is to implement a method to select the most relevant SPoC components. The fourth challenge is to implement a real-time framework using `MNE-LSL`. Finally, the last challenge is to use this toolbox to build a modular pipeline, which can run all or only some of the steps, depending on the user's needs.

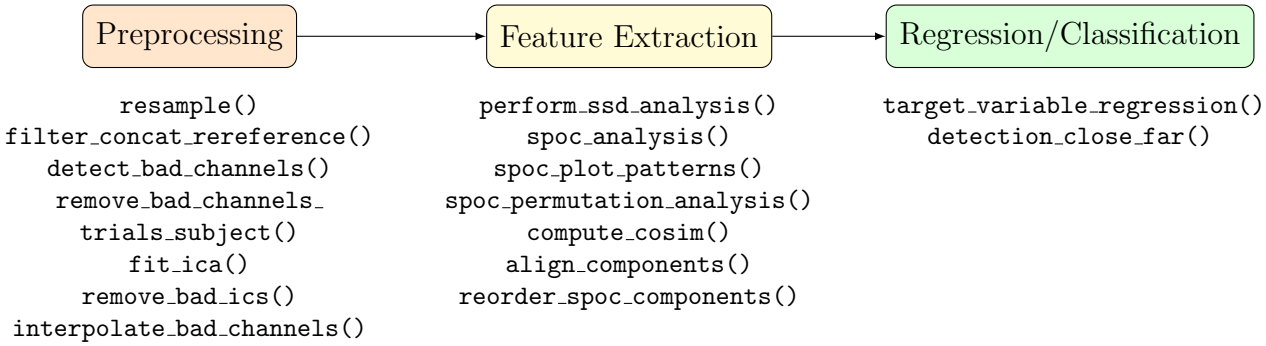


Figure 1: Diagram of the toolbox structure.

We use as much as possible existing methods, and only implement the missing ones. The main contribution of this thesis is not the methods themselves, but the way they are wrapped, combined and automated in a single toolbox.

Structure of the toolbox Figure 1 shows the three sequential stages the toolbox can perform: preprocessing, feature extraction and regression/classification. Here are the precise steps/methods implemented by this toolbox and combined into a single semi-automatic pipeline. The arguments of the functions are not given, for the sake of simplicity, and the manual steps are precised:

1. `resample()`: Resamples the EEG data to a constant sampling rate and interpolates potentially missing data.
2. `filter_concat_rereference()`: Filters, concatenates, and rereferences raw EEG data.
3. `detect_bad_channels()`: Detects and visualizes bad EEG channels, trials, or subjects.
4. `remove_bad_channels_trials_subject()`: Removes bad channels, trials, or subjects from the EEG data, based on the previous plots. The selection is done manually.
5. `fit_ica()`: Fits Independent Component Analysis (ICA) to the preprocessed EEG data. Shows plots of the independent components.
6. `remove_bad_ics()`: Removes the bad independent components interactively and saves the cleaned data. The selection is done manually.
7. `interpolate_bad_channels()`: Interpolates removed channels in the EEG data.
8. `perform_ssd_analysis()`: Performs Spatial Spectral Decomposition (SSD) analysis on the EEG data.

9. `spoc_analysis()`: Performs Source Power Comodulation (SPoC) analysis on the EEG data.
10. `spoc_plot_patterns()`: Plots SPoC activation patterns.
11. `spoc_permutation_analysis()`: Performs permutation analysis to assess the significance of the SPoC components.
12. `compute_cosim()`: Computes cosine similarity between SPoC activation patterns across subjects.
13. `align_components()`: Aligns SPoC components across subjects for consistency.
14. `reorder_spoc_components()`: Reorders SPoC components based on alignment results.
15. `target_variable_regression()`: Predicts the target variable using the selected SPoC component(s).
16. `detection_close_far()`: Classifies close position of the VR avatar vs. far position of the VR avatar (see **4.1.2 Experimental design**) using SPoC results.

Each part of the toolbox is described in detail in the next section. The realtime framework is a separated script of the toolbox—`realtime.py`—and is detailed in **3.3.9 Simulating the real-time VR experiment**.

3.3. Precise description of the toolbox

Each functionality will now be precisely described, as well as the mathematical background behind each function and algorithm used. Everything will be presented in the same order as the list above, that is the order used in the final pipeline: preprocessing, feature extraction and regression/classification.

3.3.1. Preprocessing

The following preprocessing steps are applied to the raw EEG signal:

1. Resampling & data interpolation
2. Jump artifact removal
3. Re-referencing (on average)
4. Filtering
5. Remove bad channels
6. ICA
7. Bad channel interpolation

Resampling & data interpolation

The data is sampled at a rate of 500Hz, but the time stamps are not constant, due to the wireless transmission of the mobile EEG cap. `MNE-Python` however assumes constant time windows between two consecutive timestamps. Therefore, the data is first resampled to a constant sampling rate. To do this, interpolation is performed over all data points and new data points are resampled from the interpolated curve. A cross-validation strategy is used to assess the interpolation methods and choose the best one.

Cross-validation As stated in *Tomczac et al., 1998* [54] "cross-validation (a.k.a. leaving-one-out method) is often used to select an interpolator from finite number of candidates". This article summarizes this method, as described in *Davis et al., 1987* [10]: cross validation simulates the process of predicting values at new, unmeasured times by temporarily hiding known data points, predicting their values using the remaining data, and comparing predictions to the true observed values. This provides a realistic assessment of a model's predictive accuracy. The overall performance is assessed by computing the Root Mean Square Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where n is the number of data points, y_i is the actual value (hidden), and $\hat{y}_i$ is the predicted value.

For example, cross-validation was used in practice by *Gräter et al.* [16] for interpolation on environmental data. Cross-validation allowed them to quantitatively compare different interpolation methods by withholding portions of the observed data, predicting these withheld values, and directly comparing predictions to actual measurements. This gave an objective evaluation to assess the performance of each interpolation method.

Criteria *Risk et al.* [46] compared different interpolation methods on weather data. They also used cross-validation for that, to ensure robust evaluation. More precisely, they used 10-fold cross-validation as the main validation technique to rigorously assess predictive performance. The dataset has been split into ten groups, and each group has been predicted by models trained on the remaining data. Prediction errors have been calculated using root mean squared error (RMSE) and mean absolute error (MAE).

Among all criteria used to evaluate the performance of interpolation methods as listed in the review *Lepot at al., 2017* [29], RMSE is used since it is one of the most commonly used performance metrics for interpolation. In addition, it sticks with the three previously cited articles.

Number of folds As explained in scikit-learn [40] documentation *Leave One Out (LOO)* [52]: *"As a general rule, most authors and empirical evidence suggest that*

5 or 10-fold cross validation should be preferred to LOO". Therefore, 5-fold cross-validation is used.

Interpolation methods Using this validation method, all the interpolation methods proposed by the function of `scipy` librairy `scipy.interpolate.interp1d()` are tested. This includes:

- **linear**: Interpolates linearly between the points.
- **nearest**: Returns the nearest point, rounding down when the value lies between two points.
- **nearest-up**: Returns the nearest point, rounding up when the value lies between two points.
- **zero**: Zeroth-order spline interpolation.
- **slinear**: First-order spline interpolation that performs linear interpolation between the points.
- **quadratic**: Second-order spline interpolation that uses a quadratic curve between the points.
- **cubic**: Third-order spline interpolation that uses a cubic curve between the points.
- **previous**: Interpolation using the previous point.
- **next**: Interpolation using the next point.

On Figure 2, we can see how each method look like. The result of the cross-validation, and the method chosen, are presented in **Chapter 4**. In the pipeline, the resampling/interpolation is done with `resample()`. This cross-validation doesn't have to be run on each experiment and dataset, the interpolation method chosen can be kept for every experiment and dataset. It should indeed be generalizable on every EEG dataset, with only a very small approximation error.

Jump artifact removal

Artifacts They refer to elements of the recorded signal that stem from sources unrelated to the primary focus of interest (i.e., neuronal activity in the brain) and can thus be seen as interference or noise [35]. **Environmental artifacts** include continuous oscillations at the power-line frequency and electromagnetic noise generated by nearby equipment. **Instrumentation artifacts** arise from sporadic high-amplitude fluctuations or, conversely, a stable zero signal in a single channel due to equipment failure or inadequate scalp contact. **Biological artifacts** comprise signals originating from the body, such as cardiac electrical activity, brief step-like fluctuations (notably in frontal channels) caused by eye movements, large transient shifts due to blinking, and short bursts of high-frequency activity across multiple channels associated with muscle activity during swallowing [37].

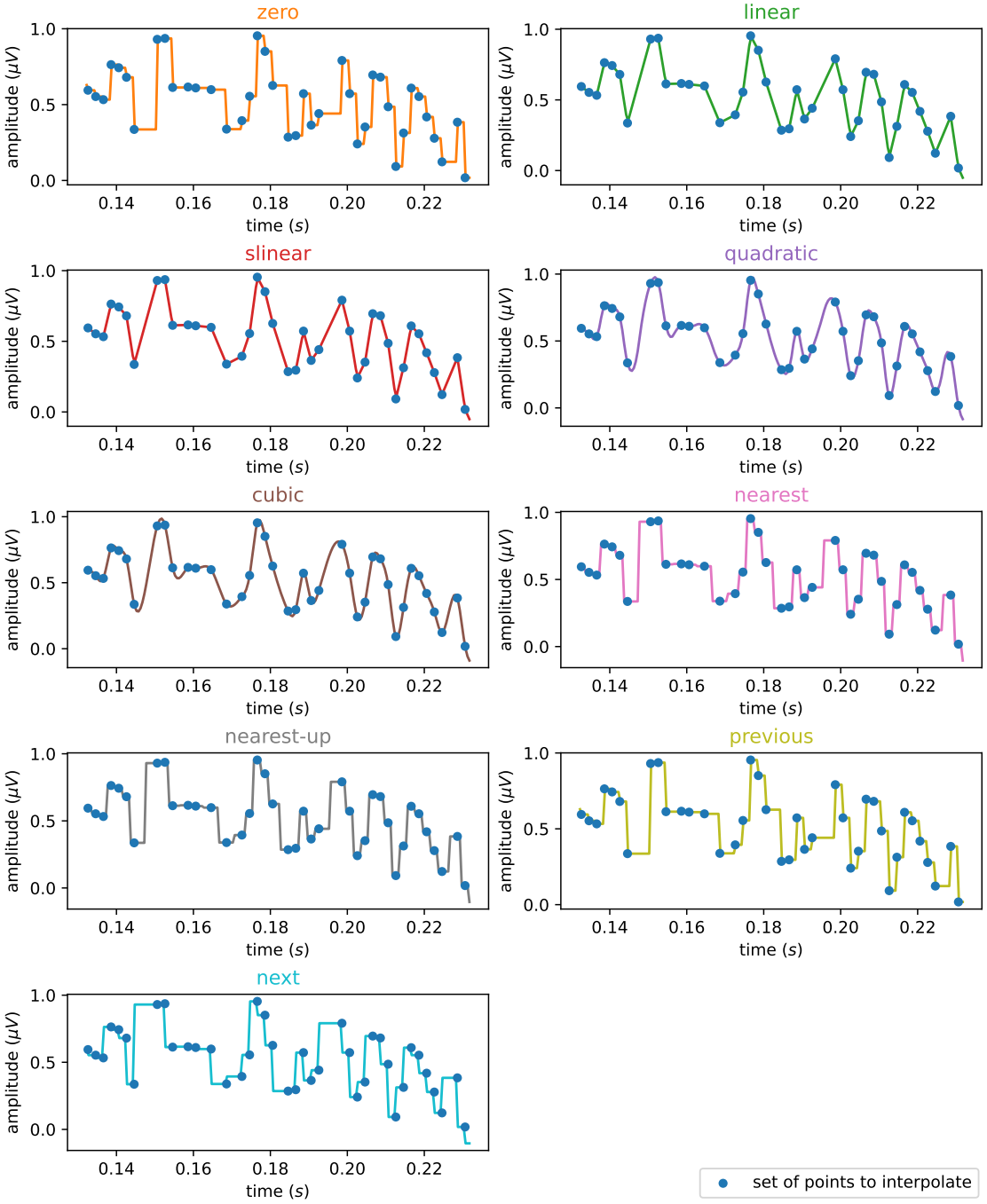


Figure 2: Different interpolation methods, given a set of points. Blue points are to interpolate, and the colored curves correspond to the interpolation.

Jump artifacts They are large, abrupt voltage changes often caused by electrode movement, disconnection, or abrupt impedance shifts—can mimic or obscure brain activity, thereby biasing both automated and visual EEG analyses. The second preprocessing step is to remove them. They are a kind of instrumentation artifact. This as well as the next two steps are performed by the function `filter_concat_rereference()`.

Removing jump artifacts from EEG recordings is crucial because without this step, artifacts may be mistaken for genuine physiological signals and lead to incorrect diagnoses or misinterpretations of neural function—for example, abrupt jumps can mimic cognitive or pathological activity and bias clinical assessments in sleep or neurological disorders [23]. Such artifacts, also degrade overall signal quality.

Re-referencing (on average)

Then, the signal is re-referenced on average.

Re-referencing EEG data to the average of all electrodes is a common practice in which the mean voltage across all channels is subtracted from each channel at every time point, as used in [21]. This method mitigates the influence of any single reference site and distributes reference-related noise or bias across the channels, thus enhancing the accuracy of signal interpretation. This is particularly important in high-density montages, such as a 64-electrode setup, where the average reference more closely approximates a *neutral* reference due to the uniform coverage of the scalp.

Artifacts from the reference electrode can affect all other channels recorded relative to it, potentially introducing bias or errors throughout the dataset. By using the average potential from all electrodes as a reference, each electrode's signal reflects its deviation from the global mean, minimizing the influence of any single electrode and facilitating a more even distribution of errors. This method is especially advantageous for source reconstruction and spatial analyses, as it reduces model error concentrated at the original reference point. Consequently, the values now represent activity relative to the global mean, ensuring consistency in comparisons between channels and preventing any one electrode location from disproportionately influencing the data.

As the density of electrodes increases, reconstruction errors are minimized, indicating that 64-channel montages significantly benefit from the average reference, resulting in lower overall bias [31].

Filtering

The signal is then filtered. Filtering EEG data is essential for enhancing signal quality and eliminating artifacts. Each filtering step plays a significant role in this process.

Bandpass filtering The first step, bandpass filtering, is designed to remove low-frequency drifts and high-frequency noise that commonly affect EEG recordings, as

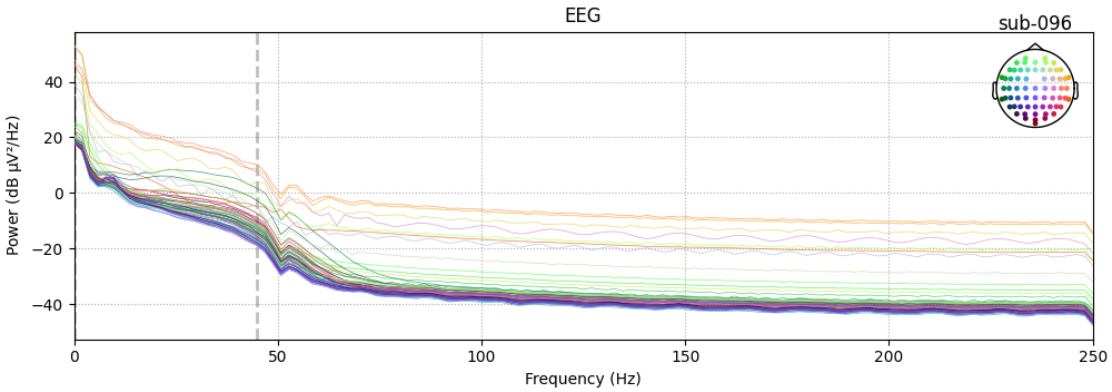


Figure 3: Power spectrum after filtering.

done in *Hofmann et al., 2021* [21]. As explained in the book *An Introduction to the Event-Related Potential Technique* [32]: "most of the relevant portion of the ERP waveform in a typical cognitive neuroscience experiment consists of frequencies between 0.01 Hz and 30 Hz". A frequency range of 0.1 to 45 Hz is chosen, as suggested in *Pieper et al., 2021* [43]. It follows the filtering recommandations of *Zhang et al., 2024* [61]. In addition, only alpha frequencies (8-13 Hz) will be looked at later.

Frequencies below 0.1 Hz can introduce slow drifts, which may result from movements or eye blinks, while frequencies above 45 Hz can encompass muscle artifacts and electrical noise. By applying bandpass filtering, the data becomes focused on the frequency ranges that are most pertinent to EEG analyses, particularly those associated with brain activity.

Power line removal The second step involves notch filtering at 50 Hz, as done in [21] and [43], which is specifically intended to eliminate power line interference. This type of noise is often present at 50 Hz (or 60 Hz, depending on the region) and can obscure genuine brain signals, potentially leading to misinterpretations of the data. By effectively removing this interference, notch filtering enhances the clarity of the EEG signal, making it easier to identify true neural activity.

Remove bad channels

The next step is to remove the bad and noisy channels. For that, the power spectrum and the maximum activity heatmap are computed for each subject. They are both essential for identifying and manually removing bad channels in EEG data. In addition, the Local Outlier Factor (LOF) score [8] is computed for each channel on each subject. This is explained in paragraph **3.3.1 LOF** in more detail. It gives an automated way to detect these bad channels and assist in the decision.

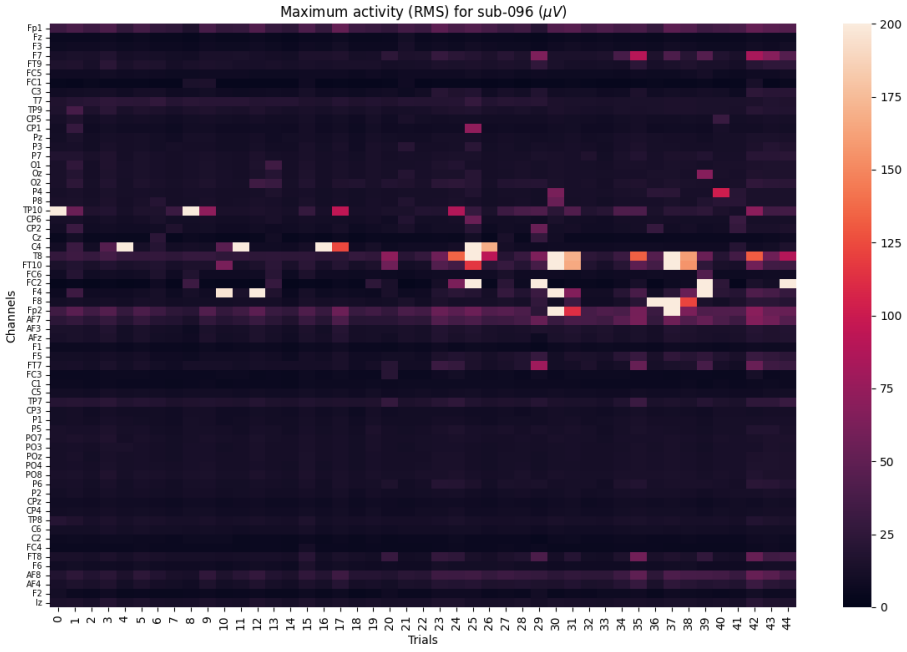


Figure 4: Maximum activity heatmap.

Power spectrum The power spectrum describes how the power of a time-domain signal is distributed across different frequencies (see Figure 3). By examining the power spectrum, one can detect channels with abnormally high or low energy in specific frequency bands—an indication of noise, line interference, loose electrodes, or amplifier saturation. Power spectral analysis can reveal channels contaminated with line noise, loose electrodes, or saturated amplifiers, making it an objective criterion for quality control.

Welch proposed a method to estimate the power spectral density (PSD) of a signal, which reduces noise in the estimated spectrum by averaging modified periodograms of overlapping segments of the signal [42]. This method is widely used in EEG analysis, due to its effectiveness in providing a reliable estimate of the power distribution across frequencies. The steps of Welch's method are summarized below:

1. Divide the signal into overlapping segments.
2. Apply a window function to each segment to reduce spectral leakage.
3. Compute the periodogram (squared magnitude of the Fourier transform) for each windowed segment.
4. Average the periodograms of all segments to obtain the final PSD estimate.

Maximum activity heatmap A maximum activity heatmap is a spatial visualization that displays, for each EEG channel, the average amplitude (Root Mean Square) recorded in a defined time window, one epoch in our case (see Figure 4).

The heatmap's color intensity reflects these peak activity values across the electrode array, enabling quick identification of channels with unusually high activity that might indicate artifacts, noise, or malfunctioning electrodes. Heatmaps are used to guide the removal of bad channels in EEG preprocessing pipelines. The Root Mean Square (RMS) is the following:

$$\mathrm{RMS}(a) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} a_i^2}$$

with a the signal of a given electrode at a given time window, N the number of timestamps in this time window, and a_i the value of the signal at timestamp i. Maximum amplitude heatmaps complement spectral analysis by offering a temporal overview of signal strength across electrodes. These heatmaps highlight channels with unusually large peaks that often correspond to artifacts or hardware faults, and they make it easier to distinguish between spatially diffuse neural activity and localized anomalies. By supplying an intuitive visual check alongside automated statistical criteria, heatmaps help ensure that only channels reflecting genuine brain signals proceed to further processing.

LOF Local Outlier Factor (LOF) is a density-based anomaly detection technique that quantifies how isolated an observation is relative to its local neighborhood [8]. In the present application to EEG bad-channel identification, each electrode is represented by a feature vector. For each channel, LOF computes a "reachability density" based on its distances to the k nearest neighbors, and then compares this density to the average reachability density of those neighbors. Channels whose local density is substantially lower than that of their neighbors—i.e. those residing in sparse regions of feature space—receive elevated LOF scores and are classified as outliers. Concretely, the algorithm proceeds in four steps:

1. Identification of the k nearest neighbors for each channel,
2. Computation of the reachability distance from the channel to each of its neighbors,
3. Estimation of the channel's local reachability density as the inverse of the mean reachability distance,
4. Calculation of the LOF score by dividing the average local reachability density of the neighbors by the channel's own reachability density. A LOF score near unity indicates that the channel's local density matches that of its neighbors, whereas scores significantly greater than one signal anomalous behavior.

Because LOF adapts to local variations in the data rather than imposing a single global threshold or distributional model, it is especially effective for bad-channel removal in EEG. Channels that exhibit only modest deviations along individual features but pronounced deviations jointly across multiple features can still be

detected. Moreover, electrodes located in different regions of the scalp—each with its own baseline activity statistics—are compared only to their most similar peers in feature space, enhancing the algorithm's sensitivity to region-specific artifacts.

As suggested in [28], a LOF threshold of 1.5 is used. It is suggested to choose the number of neighbors between 10 and 20 [8]. The maximum value 20 is chosen. It is very unlikely to have 10 bad electrodes forming a cluster (which would give all of them very low LOF scores), even in extreme cases. 20 is then impossible.

Actual removal Based on the visual inspection of the power spectrum and the maximum activation heatmap, where it is possible to see if a channel is particularly different from the others, channels to exclude are chosen, if any. The LOF scores assist in this decision, which can identify outliers that may not be obvious with visual inspection. The LOF score approach is however not robust enough to be used alone automatically: it sometimes gives high scores to non-problematic channels, or low ones to problematic ones.

These steps are performed by the functions `detect_bad_channels()` and `remove_bad_channels_trials_subject()`.

ICA

Jump artifacts have been removed previously, but there are many other artifacts contaminating the signal.

Latent sources Independent Component Analysis (ICA) is a type of Blind Source Separation technique. It is used to separate a multivariate signal into additive, independent components. ICA decomposes the data $\mathbf{X} \in \mathbb{R}^{C \times T}$ as follows:

$$\mathbf{X} = \mathbf{A}\mathbf{s}$$

where $\mathbf{A} \in \mathbb{R}^{C \times C}$ is the mixing matrix and $\mathbf{s} \in \mathbb{R}^{C \times T}$ represents the (independent) latent sources, as explained in [36]. Each row of $\mathbf{s}$ is a statistically independent time series, and each column of $\mathbf{A}$ acts as a spatial filter. To recover the sources, we can apply the inverse of the mixing process. Assuming we have estimated the mixing matrix $\mathbf{A}$ and the observed data $\mathbf{X}$, we can recover the independent sources $\mathbf{s}$ using:

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{X}$$

This allows us to obtain the latent sources from the observed data.

Assumptions It is particularly useful in EEG data analysis for isolating and removing artifacts from the recorded brain signals. The fundamental assumption of ICA is that the observed signals are linear mixtures of statistically independent source signals. We assume it is the case for our EEG signals.

In addition, it has been shown that applying a high-pass filter (1-2 Hz) before performing ICA improves the decomposition [60]. Therefore, the EEG data is copied and a 1 Hz high-pass filter is applied before the ICA, to get the independent

components. For the rest of the analysis, the EEG data before the copy is used. The 1 Hz high-pass filter is only used to improve the ICA decomposition.

Actual removal By applying ICA, the EEG data can be decomposed into these independent components, some of which correspond to artifacts such as eye blinks, muscle activity, or heartbeats. Once identified, these artifacts are removed by subtracting them from the global signal, allowing for a cleaner representation of the underlying neural activity. It has been shown in the literature that removing artifacts after an ICA significantly improves the signal-to-noise ratio [48]. The fastICA implementation as proposed by MNE-Python [17] is used.

These steps are performed by the functions `fit_ica()` and `remove_bad_ics()`.

Bad channel interpolation

MNE-Python repairs bad EEG channels using spherical spline interpolation [41]: electrode positions are mapped to a unit sphere, and each bad channel's signal is estimated from the signals at the remaining good channels. This leverages the spatial smoothness of scalp potentials to infer plausible values at bad sensors from their neighbors.

A precise description of how it is done, as described in MNE documentation *Algorithms and other implementation details* [34] is given in Appendix A.

This step is performed by the function `interpolate_bad_channels()`. This is the last preprocessing step, before extracting features and running the analysis.

3.3.2. Spatio-Spectral Decomposition

The aim of Spatio-Spectral Decomposition (SSD) is to extract neural oscillations from a given frequency band given multi-channel EEG recordings [39]. In the use case validation presented in **Chapter 4**, the frequency band of interest is the alpha band. It also allows for dimensionality reduction. It does it by increasing the power of the peak frequency—the frequency of interest—, and decreasing the power of the surrounding frequency bands—noise (Figure 5). It thus increases the signal-to-noise ratio (SNR). It is achieved through a generalized eigenvalue decomposition. The results are a spatial filter as well as patterns, which highlight the areas generating oscillations in the alpha band, as opposed to areas generating noise.

The first thing to compute, before running SSD, is each participant's individual peak of alpha frequency band. It is not always 10 Hz, and varies among participants. This is done by:

1. computing the power spectrum with Welch's method [42],
2. disentangling the power of the $1/f$ aperiodic signal from the power of the alpha frequency band, using the FOOOF method [12] (Fitting Oscillations & One-Over-F).

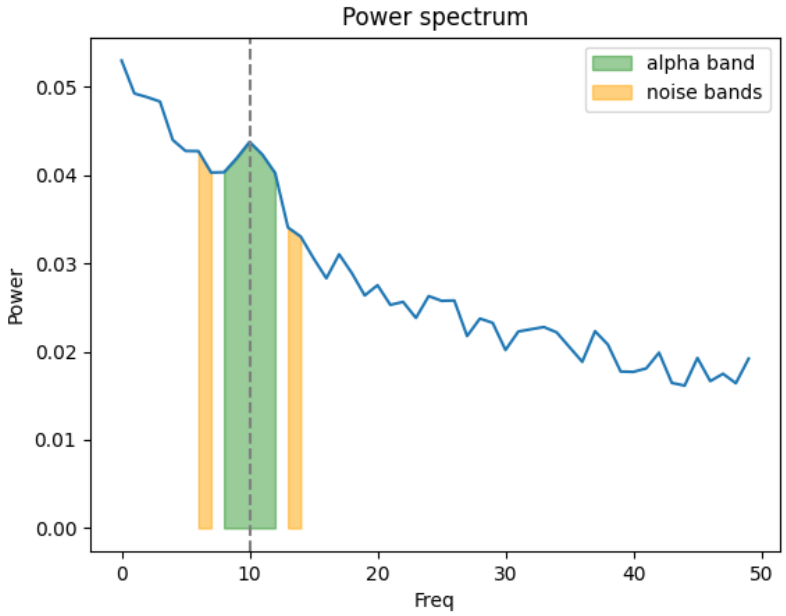


Figure 5: Illustration of the adapted SSD frequency bands. The target band is centered around the individual alpha peak and has a width of 4 Hz. The two flanking bands have a width of 2 Hz and are separated from the target band by a gap of 1 Hz.

By this way, we get the peak alpha frequency. We can now define our alpha band and the flanking bands. Following what has been done in *Hofmann et al.* [21], a 1 Hz gap is inserted between the alpha frequency bands and the surrounding frequency bands (Figure 5). Noise is likely to be present, but our signal of interest may also remain there. The SSD method will neither decrease it, nor enhance it.

We then run SSD, given the frequency bands. To dive more precisely into the method, SSD uses a linear decomposition approach where the recorded signal is modeled as the sum of the desired signal, and noise, expressed as:

$$\mathbf{M} = \mathbf{S} + \mathbf{N} \tag{3.1}$$

where M is the measured activity, S is the signal of interest, and N is the noise.

It then extracts the signal at different frequency band: matrix $\mathbf{M}_s$ for the target frequency band (global signal $\mathbf{M}$ filtered in the target band) and matrix $\mathbf{M}_n$ for the flanking bands (global signal $\mathbf{M}$ filtered in the flanking bands), and it computes the time-averaged covariance matrices for both filtered signals:

$$\mathbf{C}_s = \frac{\mathbf{M}_s^\top \mathbf{M}_s}{t} \quad \text{and} \quad \mathbf{C}_n = \frac{\mathbf{M}_n^\top \mathbf{M}_n}{t} \tag{3.2}$$

SSD aims at finding spatial filters that maximize the ratio:

$$\text{SNR}(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{C}_s \mathbf{w}}{\mathbf{w}^\top \mathbf{C}_n \mathbf{w}} \tag{3.3}$$

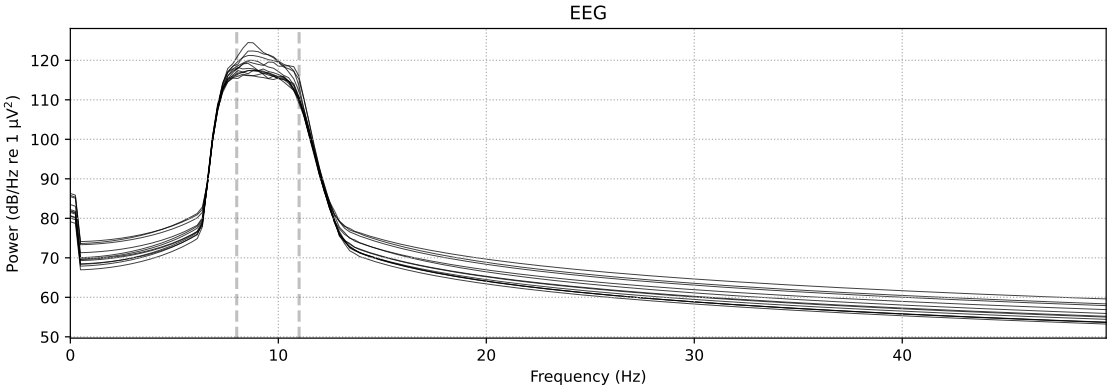


Figure 6: Power spectrum of the signal after SSD processing, for all SSD components of subject 96. We can observe that only the target frequency band is kept.

This maximization is achieved through generalized eigenvalue decomposition:

$$\mathbf{C}_s\mathbf{w} = \lambda\mathbf{C}_n\mathbf{w} \tag{3.4}$$

leading to filters that accentuate the alpha oscillatory activity while weakening noise.

We now have to select the SSD components, from the 64 the model outputs. We only retain components with enough alpha information as compared to noise. Quantitatively, we compute spectral ratios:

$$\text{Spectral Ratio} = \frac{P_{\text{alpha}}}{P_{\text{flanking}}} \tag{3.5}$$

where P_{alpha} is the mean spectral power in the target frequency range, and P_{flanking} the combined mean spectral power in the flanking frequency ranges. We sort the component according to their ratio. We then compute the elbow point - the point of maximum curvature - with the kneedle algorithm [49]. It is the point where the spectral ratio starts to flatten out, indicating that the noise starts to dominate. We keep all components before this point. We keep a minumum of 10 components, even if the elbow point is below, to still get enough signal.

The final step of SSD is to filter our signal in the alpha frequency range, to only keep these frequencies. The result is a set of neural oscillations in the alpha band (see Figure 6), with an increased SNR.

As a side note, it is to mention that we could directly have band-pass filtered our signal in the alpha frequencies, rather than applying SSD. However, SSD allows for dimensionality reduction, and increases the SNR. It indeed also removes noise *within* the alpha frequency band, and is thus a more robust method.

This whole spatio-spectral decomposition step is performed by the function `perform_ssd_analysis()`.

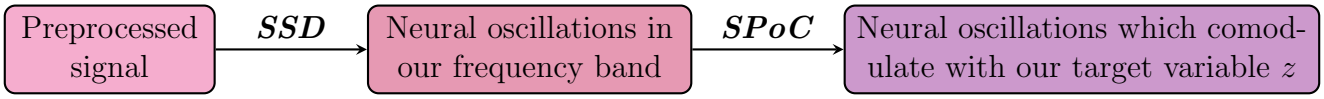


Figure 7: SSD and SPoC framework.

3.3.3. Source Power Comodulation

We now want to extract the neural oscillations that comodulate with a target variable $\mathbf{z}$. As an example, $\mathbf{z}$ will be the distance of the avatar to the participant over time, in the use case validation which will be presented later.

SPoC after SSD To extract these neural oscillations, the Source Power Comodulation framework—SPoC (SPoC$_\lambda$ algorithm) [13]—is used. It is a supervised source separation framework. Unlike ICA, it makes use of the target variable $\mathbf{z}$. It is based on the idea that neuronal components that have a high covariance in power with $\mathbf{z}$ can be extracted. The result is a set of spatial filters $\mathbf{W}$ which optimize the comodulation between $\mathbf{z}$ and the spatially filtered signal.

The input of the SPoC are the selected SSD component. An SSD is run first because it enhances the power of the frequency of interest, and reduces the noise around it. Then SPoC uses this improved signal along with the target variable $\mathbf{z}$ to extract the neural oscillations associated to it. The method is summed up on Figure 7.

SPoC$_\lambda$ algorithm The signal is first epoched. A set of overlapping $1s$-EEG epochs is obtained from the SSD-filtered data ($0.5s$ overlapping). This duration of $1s$ is chosen based on *Hofmann et al., 2021* [21]. The overlap is chosen to smooth the analysis and to avoid giving more importance to the center of the epoch, as compared to the edges [53]. The second algorithm (SPoC$_\lambda$) from the Source Power Comodulation framework is then applied. The first one is not used since there isn't any analytical solution to it. SPoC$_\lambda$ allows to be sure to find the global minimum, quickly, and in an interpretable way. It works as the following. As explained in *Dähne et al., 2014* [13], we have an objective function:

$$f_\lambda = \mathrm{Cov}\left[\tilde{\mathbf{z}}(e), \mathbf{z}(e)\right] = \mathbf{w}^\top \mathbf{C}_z \mathbf{w},$$

and a norm constraint:

$$\mathrm{Var}\left[\mathbf{w}^\top \mathbf{x}(t)\right] = \mathbf{w}^\top \mathbf{C} \mathbf{w} \stackrel{!}{=} 1.$$

where $\mathbf{z}(e)$ is the target variable in epoch e, $\tilde{\mathbf{z}}(e)$ is the estimation of $\mathbf{z}$ in epoch e, $\mathbf{w}$ is the spatial filter we want to find, $\mathbf{C}(e)$ is the covariance matrix in epoch e, $\langle\cdot\rangle$ is the average over all epochs, $\boldsymbol{\mathcal{C}}_z$ is the weighted covariance matrix defined as $\boldsymbol{\mathcal{C}}_z = \langle \mathbf{C}(e)\mathbf{z}(e)\rangle$, and $\boldsymbol{\mathcal{C}}$ is the mean covariance matrix defined as $\boldsymbol{\mathcal{C}} = \langle \mathbf{C}(e)\rangle$.

We can then solve this with the Lagrangian multiplier method. It leads to a generalized eigenvalue problem:

$$\boldsymbol{\mathcal{C}}_z \mathbf{w} = \lambda \boldsymbol{\mathcal{C}} \mathbf{w},$$

where λ is the eigenvalue corresponding to the eigenvector/component $\mathbf{w}$. It represents the covariance between $\mathbf{z}$ and $\tilde{\mathbf{z}}$, that is how much the associated component comodulates with $\mathbf{z}$.

The eigenvectors obtained are stored in a matrix $\mathbf{W}$. They are the spatial filters $\mathbf{W}$ we need. The eigenvectors are sorted with respect to their corresponding $|\lambda|$, in decreasing order.

Activation patterns The spatial filters we get from SPoC aren't directly interpretable as activation of the sources on the brain [19]. We need to compute activation patterns from the filters, which are the spatial patterns of the sources we can directly interpret. They are obtained by correcting the covariance of the filters, as follows:

$$\mathbf{A} = \mathbf{\Sigma}_x \cdot \mathbf{W} \cdot \mathbf{\Sigma}_{\tilde{s}}^{-1}$$

where $\mathbf{A}$ represents the activation patterns, $\mathbf{W}$ the spatial filters, $\mathbf{\Sigma}_x$ the covariance matrix of the data, and $\mathbf{\Sigma}_{\tilde{s}}$ the covariance matrix of the SPoC components. It can usually be simplified when the SPoC components are uncorrelated. However, it is not exactly the case here: $\mathbf{\Sigma}_{\tilde{s}} \neq I$. We will thus stick with this formula to compute them.

The SPoC components and activation patterns are extracted with `spoc_analysis()` while the activation patterns are plotted with `spoc_plot_patterns()`.

3.3.4. Pearson correlation coefficient

As a metric to evaluate the performance of the method, the Pearson correlation coefficient r between the target variable $\mathbf{z}$ and its estimation $\tilde{\mathbf{z}}$ have been used. It is a common metric to evaluate the linear association between two variables, as done in *Hofmann et al., 2021* [21].

The Pearson correlation coefficient is a measure of the linear relationship between two datasets, for example $\mathbf{z}$ and $\tilde{\mathbf{z}}$. The formula for the Pearson correlation coefficient r is:

$$r = \frac{\sum_{i=1}^{n}(\mathbf{z}_i - \bar{\mathbf{z}})(\tilde{\mathbf{z}}_i - \bar{\tilde{z}})}{\sqrt{\sum_{i=1}^{n}(\mathbf{z}_i - \bar{\mathbf{z}})^2}\sqrt{\sum_{i=1}^{n}(\tilde{\mathbf{z}}_i - \bar{\tilde{z}})^2}}$$

where $\bar{\mathbf{z}}$ and $\bar{\tilde{z}}$ are the respective means of $\mathbf{z}$ and $\tilde{\mathbf{z}}$, and n is the number of data points.

The value of r can range from -1 to $+1$, where $r = 1$ indicates perfect positive correlation, $r = -1$ indicates perfect negative correlation, and $r = 0$ indicates no linear correlation. A positive correlation indicates that as one variable x increases, the other variable y also increases, while negative values show that y decreases as x increases (Figure 8).

This works well to assess the target variable prediction given a component. We however have to know which one(s) to choose between all the components. This is what follows, with different methods and approaches. It is also to be mentioned that a fixed number C of SPoC components will be kept for further analysis.

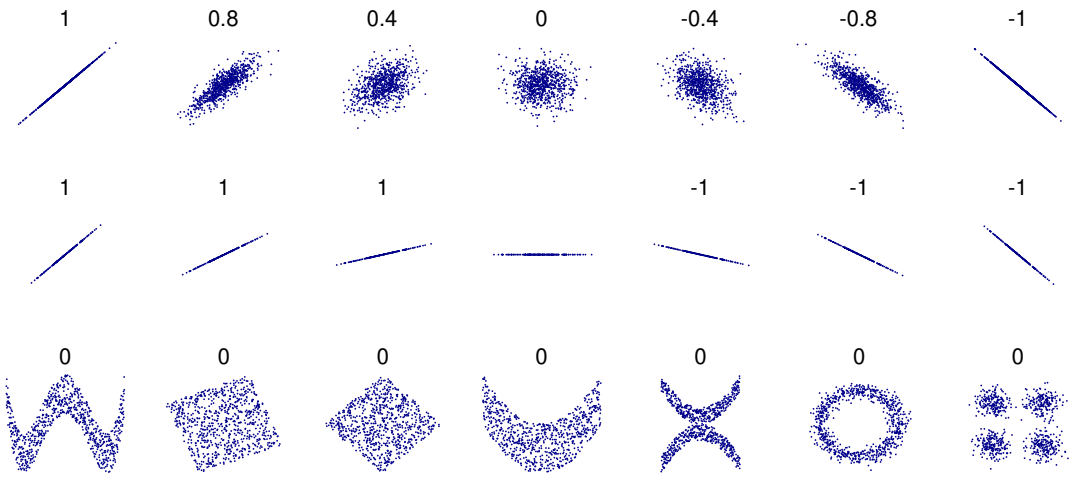


Figure 8: Correlation examples (from *wikipedia.org*).

3.3.5. Permutation analysis

The primary purpose of permutation analysis is to determine the stability and statistical significance of the relationships identified through SPoC. It helps in validating whether the observed comodulations between EEG alpha power and emotional arousal are genuine or could have arisen by chance. We implement the procedure (Figure 9) as described in *Hofmann et al.* [21]:

1. The original target variable $\mathbf{z}$ is shuffled multiple times (n=1000) to create bootstrapped data $\mathbf{z}_{\text{shuffled}}$ with randomized relationships.

2. The SPoC algorithm is applied to both $\mathbf{z}$ and $\mathbf{z}_{\text{shuffled}}$, along with the SSD components (these are never shuffled). The alpha power components are extracted based on their comodulation with the target variable.

3. The SPoC spatial filters $\mathbf{w}$ (resp. $\mathbf{w}_{\text{shuffled}}$) are then used to make a prediction of the target variable $\mathbf{z}$ (resp. $\mathbf{z}_{\text{shuffled}}$), for each component, using the following formula (as already seen in 3.3.7):

$$\tilde{\mathbf{z}}(e) = \mathbf{w}^{\top} \left(\mathbf{C}(e) - \boldsymbol{\mathcal{C}} \right) \mathbf{w} \tag{3.6}$$

$$\tilde{\mathbf{z}}_{\text{shuffled}}(e) = \mathbf{w}_{\text{shuffled}}^{\top} \left(\mathbf{C}(e) - \boldsymbol{\mathcal{C}} \right) \mathbf{w}_{\text{shuffled}} \qquad (\times 1000) \tag{3.7}$$

4. The Pearson correlation coefficient between the EEG-derived target variable computed for each epoch $\tilde{\mathbf{z}}$ (resp. $\tilde{\mathbf{z}}_{\text{shuffled}}$) and $\mathbf{z}$ (resp. $\mathbf{z}_{\text{shuffled}}$) is calculated, for each component.

We obtain 1000 sets of C correlations $r_{\text{shuffled}} = [r_0, ..., r_C]$, one correlation for each component, and one set for each permutation $\mathbf{z}_{\text{shuffled}}$. It yields a distribution of correlations per component. We also obtain r, the correlations for our non-permuted target variable vector $\mathbf{z}$. If the correlation r is significantly better than

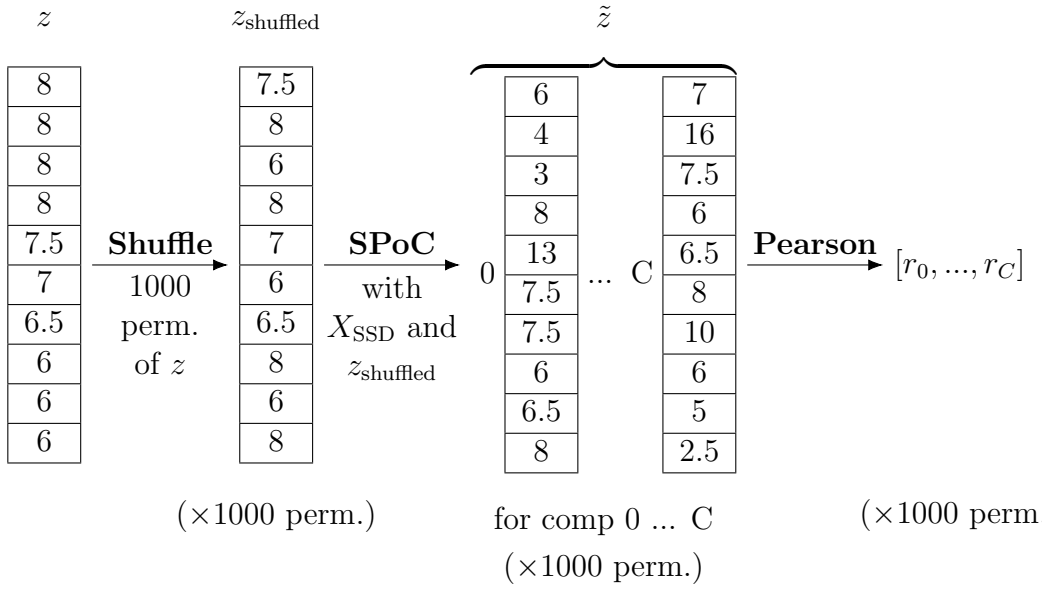


Figure 9: Schema of the permutation analysis procedure.

the permuted correlations, it means that the relationship between alpha power and emotional arousal is statistically significant and not a result of random fluctuations.

To assess the significance of our result, we sort the correlations r_{shuffled} (per component) and get the correlation r_{thres} so that 95% of them are under it. We compare it to the correlation r obtained on $\mathbf{z}$ without shuffling. If $r > r_{thres}$, our result is significant and our component is useful. It is unlikely to be due to random chance. If $r < r_{thres}$, our component may rely on spurious correlations or noise to make the prediction and is therefore not trustworthy. We can thus see, for a given subject, which of its components are significant and which are not.

This step is performed by the function `spoc_permutation_analysis()`.

3.3.6. SPoC component selection

Hofmann et al. [21] shows that the activation patterns should be located in the parieto-occipital regions, given our experiment with emotional arousal. We thus need a way to select the best patterns, those containing our signal of interest, among the significant ones—each pattern is linked to a component, thus selecting a pattern also selects a component. There will indeed remain noise in the activation patterns. Three approaches are proposed here: one single-subject and two across-subjects.

Approach 1 (single-subject): Select prototypes

The paper *Harmeling et al., 2006* [18] proposes a method to select prototypes among the activation patterns, which are the most representative of the data. It is based on the idea that we can select the patterns that are most similar to each other, and that the patterns of our signal of interest will be close to each other.

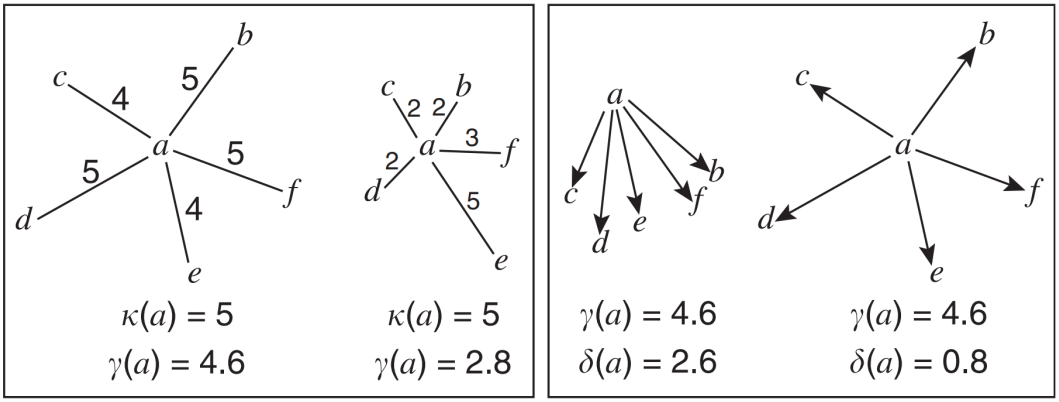


Figure 10: Illustration of the three prototype selection methods, as shown in *Harmeling et al., 2006* [18].

We first have to compute the k-nearest neighbors of each pattern, referred to here as a vector. We compute that in the sense of the euclidian norm $||\mathbf{x}|| = \sqrt{\mathbf{x}^\top \mathbf{x}}$.

We then, for each vector, order all the other vectors in ascending order of distance. It allows us to retrieve its k-nearest neighbors (kNN). Three selection methods are then proposed:

- **K-nearest neighbor** κ: The score is the distance from the vector to its k^{th} nearest neighbor:
$$\kappa(\mathbf{x}) = ||\mathbf{x} - \mathbf{z}_k(\mathbf{x})||$$
where $\mathbf{x}$ is the pattern/vector we consider, and $\mathbf{z}_k(\mathbf{x})$ is the k^{th} nearest neighbor of $\mathbf{x}$.

- **Averaged distance to kNN** γ: The score is the average of all the distances of the kNN:
$$\gamma(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} ||\mathbf{x} - \mathbf{z}_i(\mathbf{x})||$$

- **Length of the kNN vector mean** δ: We compute the mean of the k-nearest neighbor vectors in vector space, and then take its norm. When a vector is an outlier, its kNN will point in the same direction and the score will be high, whereas when a vector is a prototype, its kNN will point in different directions and they will cancel each other in mean:
$$\delta(\mathbf{x}) = \left|\left|\frac{1}{k}\sum_{i=1}^{k}(\mathbf{x} - \mathbf{z}_i(\mathbf{x}))\right|\right|$$

According to the paper, the **averaged distance to kNN** (which is refered to as γ) is a refined version of the **k-nearest neighbor** (which is refered to as κ), so method κ is not implemented in the toolbox. Method γ and the **length of the**

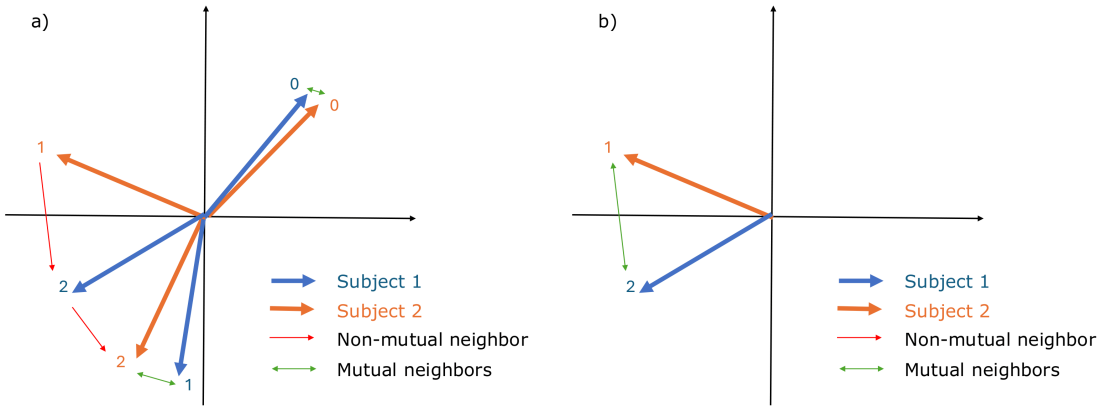


Figure 11: Mutual and non-mutual matches, among patterns in 2D vector space. On the left, we have 4 vectors that find a mutual match, and 2 vectors that don't. On the right, after removing the mutual matches, our two remaining vectors now mutually match.

kNN vector mean (which is refered to as δ) are implemented. The differences between these methods are illustrated in Figure 10.

- γ **method**: The k-nearest neighbors of each activation pattern $\mathbf{v}$ of a subject are selected. The distance used is the cosine similarity: $\text{cosim}(\mathbf{v}, \mathbf{u}) = \frac{\mathbf{v} \cdot \mathbf{u}}{||\mathbf{v}|| \cdot ||\mathbf{u}||}$, since we are interested in the direction of the vectors. Vectors with similar direction should be clustered together. The average of the cosine similarities between $\mathbf{v}$ and its kNN is then computed. A high score means that $\mathbf{v}$ has a similar direction to its kNN.

- δ **method**: The k-nearest neighbors of each activation pattern $\mathbf{v}$ of a subject (the distance is here the euclidian distance between the two normalized vectors) are selected. The mean vector of the difference between $\mathbf{v}$ and its kNN is then computed, and the norm is taken. A low score means that $\mathbf{v}$ is central to its kNN, which is a good proprety for a prototype.

Approach 2 (across-subjects): Align over one baseline subject

The goal for this approach is to align the activation patterns across subjects. More than finding good prototypes in each subject, one would be interested by linking the selected prototypes across subjects. The good patterns should be similar and present across all subjects. We would then just have to select the good ones for one subject, align the other subjects on this one, and retain the same patterns on all the other subjects. The challenge here is that the order of the patterns is not the same across subjects. They are indeed ordered by the absolute value of their eigenvalues λ, which are subject-specific.

To compare SPoC activation patterns across subjects, an alignment algorithm based on pairwise cosine similarity (Algoritm 1) is implemented. For each subject,

Algorithm 1 Align SPoC activation patterns

for each subject A_1 **do**
  for each subject A_2 **do**
    if $A_1 \neq A_2$ **then**
      $a_1 \leftarrow$ activation patterns for subject A_1
      $a_2 \leftarrow$ activation patterns for subject A_2
      $C = \text{cosine\_similarity}(a_1^T, a_2^T)^T$
      $pairs \leftarrow$ empty list
      $n \leftarrow$ number of SPoC components
      while length of $pairs < n$ **do**
        $closest_1 \leftarrow$ arg max of C along axis 0
        $closest_2 \leftarrow$ arg max of C along axis 1
        for $i = 1$ to n **do**
          $j \leftarrow closest_1[i]$
          if $closest_2[j] == i$ and $(i, j) \notin pairs$ **then**
            Add (i, j) to $pairs$
            Remove row i and column j from C
          end if
        end for
      end while
    end if
  end for
end for
return $pairs$

the set of activation patterns was compared to those of every other subject. For each pair of subjects, the algorithm computes a cosine similarity matrix between their components. One column i of the matrix represents the cosine similarities between component i of the first subject, and all the components of the second subject. It then iteratively matches pairs of components that are mutual nearest neighbors (i.e., pairs (component, counterpart) so that component has the highest cosine similarity to its counterpart in the other subject, and vice versa; it however isn't always the case, see Figure 11). We call that a mutual match. Once a pair is mutually matched, the corresponding row and column in the cosim matrix are zeroed out, to prevent repeated matches. This process continues until all components are paired, resulting in a set of matched components that maximize direction similarity across subjects.

This approach ensures that the comparison of SPoC components is robust to permutation and ordering differences, facilitating group-level analyses of spatial patterns.

We are ensured this algorithm terminates, since a real matrix of finite dimension has a maximum. The cosim matrix $\mathbf{C}$ then has a maximum $c_{j,i}$. This maximum is in particular the maximum over column i and the maximum over row j. Since the nearest neighbor of component i in subject A_1 is the arg max of $\mathbf{C}_{:,i}$ (column i of

$\mathbf{C}$), and the nearest neighbor of component j in subject A_2 is the arg max of $\mathbf{C}_{j,:}$ (row j of $\mathbf{C}$), j is the nearest neighbor of i and i is the nearest neighbor of j. They thus mutually match. There is at least one mutual match at each iteration. Since they are removed after the match, we can even say there is at least one new mutual match (because new maximum) at each iteration. The algorithm thus terminates.

Approach 3 (across-subjects): Align without baseline subject

Problem Overview We have a set of 10×10 cosine-similarity matrices, each representing the similarity between components for every pair of subjects. The goal is to find a single, consistent ordering of the 10 components for each subject, without relying on a predefined baseline. To achieve this, the idea is to fuse all pairwise similarities using spectral synchronization and then refine each subject's block with the Hungarian algorithm.

Affinity Matrix ($\mathbf{Q}$) The affinity matrix, denoted as $\mathbf{Q}$, is a $(dN) \times (dN)$ matrix, where N is the total number of subjects and d the number of SPoC components.

$$\mathbf{Q} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \cdots & \mathbf{S}_{1N} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \cdots & \mathbf{S}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{N1} & \mathbf{S}_{N2} & \cdots & \mathbf{S}_{NN} \end{bmatrix}$$

Each block $\mathbf{S}_{ij}$ is a $d \times d$ matrix representing the symmetrized similarity between subject_i and subject_j, with values in $[0, 1]$ to ensure they are normalized and bounded.

Permutation Synchronization Process To synchronize the permutations, we first symmetrize $\mathbf{Q}$ and add a small amount of jitter to its diagonal. It is a standard trick in spectral methods to ensure numerical stability, avoid trivial solutions, and improve the robustness of the eigendecomposition. The eigenvector computation can become numerically unstable if the matrix $\mathbf{Q}$ has repeated or very close eigenvalues. This often happens when the matrix is symmetric and has a block structure, as in our case. Without jitter, the eigendecomposition might not converge or could produce inconsistent results across runs. Next, we compute the top d eigenvectors of $\mathbf{Q}$ using the eigenvalue equation:

$$\mathbf{Q}\,\mathbf{v}_k = \lambda_k\,\mathbf{v}_k, \quad \text{for } k = 1, \ldots, d.$$

These eigenvectors of size $dN \times 1$ are then stacked into a matrix $\mathbf{V}$ of size $(dN) \times d$.

$$\mathbf{V} = \begin{bmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_d \\ | & & | \end{bmatrix}$$

They capture the dominant, shared patterns of similarity across all subjects, by embedding each subject's 10 components into a shared space. Each eigenvector

can be seens as aligning similar components across different subjects. To give an intuition on why this works: the eigenvectors of $\mathbf{Q}$ are derived from its spectral decomposition, which reveals the principal axes of variation in the data. $\mathbf{Q}$ is constructed such that its blocks encode pairwise similarities between subjects. The eigenvectors of $\mathbf{Q}$ naturally exploit this structure to identify components that are consistently similar across subjects.

Finally, $\mathbf{V}$ is split into $d \times d$ blocks $\mathbf{Y}_i$, each corresponding to a subject:

$$\mathbf{Y}_i = \begin{bmatrix} v_{i.d,1} & v_{i.d,2} & \cdots & v_{i.d,d} \\ v_{i.d+1,1} & v_{i.d+1,2} & \cdots & v_{i.d+1,d} \\ \vdots & \vdots & \ddots & \vdots \\ v_{i.d+(d-1),1} & v_{i.d+(d-1),2} & \cdots & v_{i.d+(d-1),d} \end{bmatrix}$$

Hungarian Algorithm Application For each subject i, we extract the block $\mathbf{Y}_i$ from $\mathbf{V}$. The rows of $\mathbf{Y}_i$ are normalized to have unit norm. we then solve the assignment problem:

$$\max_{\mathbf{P} \in \{0,1\}^{d \times d}} \langle \mathbf{P}, \mathbf{Y}_i \rangle,$$

subject to the constraints that $\mathbf{P}$ is a permutation matrix. This is achieved by applying the Hungarian algorithm [27] to the cost matrix $-\mathbf{Y}_i$.

This step of Hungarian algoritm is crucial since eigenvectors for each subject in the shared space are not yet aligned to a common reference. Each subject's components could be permuted in any order. It ensures that the ordering of components for each subject is consistent with the global patterns.

Final Mapping and Output The results are assembled into a $d \times N$ table, where each row represents a global pattern (from 0 to $d-1$), and each column corresponds to a subject. The cells in the table indicate which component index matches each global pattern for that subject. This table is saved as a CSV file.

Only this approach is kept in the toolbox: it is the most robust one among the three (see **Chapter 4 - Validation**). This step is performed by the functions `compute_cosim()`, `align_components()` and `reorder_spoc_components()`.

3.3.7. Target variable regression

We can predict the target variable $\mathbf{z}$ for each epoch e, for a given SPoC component $\mathbf{w}$. The prediction formula is the following [13]:

$$\tilde{\mathbf{z}}(e) = \mathbf{w}^\top (\mathbf{C}(e) - \boldsymbol{\mathcal{C}})\, \mathbf{w}.$$

with $\mathbf{w}$ a column of the matrix $\mathbf{W}$ of spatial filters obtained by SPoC.

3.3.8. Detection of events

This part depends on the specific task we problem to address. It is presented for the specific experiment which will be described later, in **4.1.2 Experimental design**. It can however be adapted to different use cases.

For the example of our case study, we want to detect when the avatar is in a *close* phase or a *far* phase from the subject, based on the target variable regression $\tilde{\mathbf{z}}(e)$. By *far* we mean standing far from the subject (before the approach phase), and by *close* we mean standing close to the subject (after the approach phase). We use for that the function `find_peaks()` from `scipy.signal` to find peaks and troughs in the predicted target variable $\tilde{\mathbf{z}}(e)$. We set a minimum time between two peaks/troughs of 10s, to avoid detecting too many of them close to each other.

We then evaluate the accuracy by checking if the time stamp of each detected peak/trough is indeed in a *close* or *far* phase, thanks to the ground truth target variable. The accuracy is computed as the ratio of correctly classified peaks/troughs over the total number of *close* or *far* phase.

This is done by the function `detection_close_far()`.

For the personal space intrusion detection, we want to detect when the avatar enters the personal space of the subject. From the target variable regression, we set a distance threshold, and predict all distance values under it as PS intrusion. This threshold is determined by looking at the area under the ROC (receiver operating characteristic) curve, and finding a balance between true positives and false positives. This is performed by the function `detection_ps_intrusion()`.

3.3.9. Simulating the real-time VR experiment

In parallel, a framework to mimic realtime processing is implemented, from offline EEG data stored in XDF files.

Framework The first step is to simulate the real-time stream. 3 LSL *StreamOutlet* are set up to push data at regular intervals - 2ms (the sampling rate of the EEG data is 500Hz):

- The first *StreamOutlet* (an LSL object) pushes EEG data,
- The second *StreamOutlet* gives the distance of the avatar from the subject,
- The third *StreamOutlet* gives event markers ('AvatarStartsWalking', 'PersonalSpaceEntered'...).

Every 2ms, one EEG data sample is pushed to the stream, for all 64 channels. The two other streams have fewer data, available at irregular intervals. My approach is to align the reference timestamps of the EEG data with the other two streams, and then to push data from the two streams according to the newly aligned timestamps. We create one independent thread for each stream, to mimic what we would have in a real experiment (an EEG cap and a VR headset pushing data every 2ms). We now have 3 independent *StreamOutlets* in their respective threads, pushing data simultaneously (see Figure 12). This recreates the condition of the real experiment, as if the streams directly came from the EEG cap and the VR headset.

The second step is to connect to these streams to pull the data at regular intervals. For that, we use 3 *StreamInlet* (another LSL object). Each one is connected to its corresponding *StreamOutlet*, as it would connect to the EEG stream or the

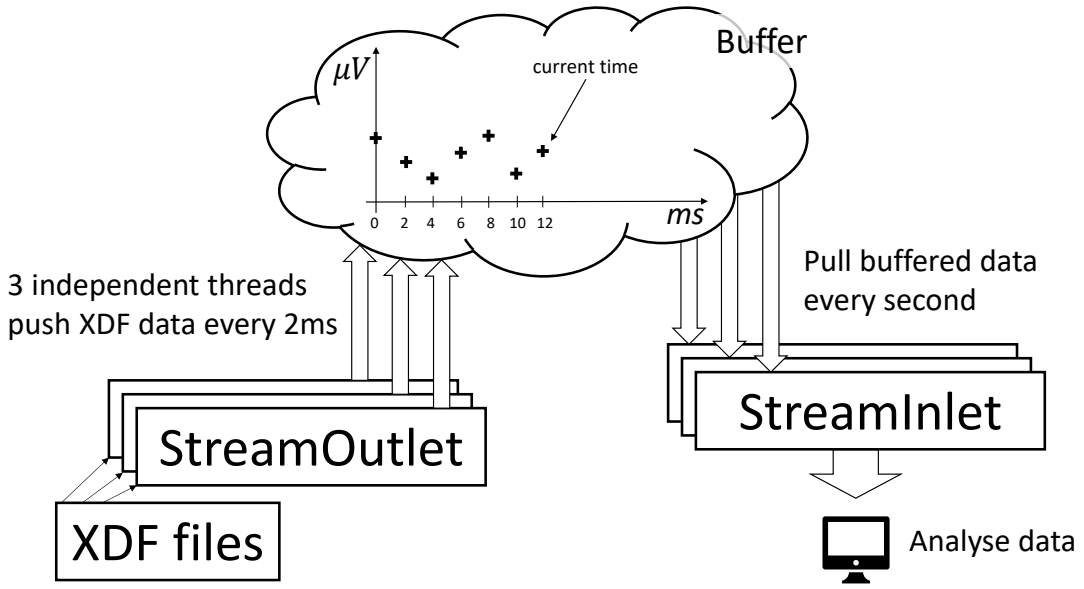


Figure 12: Simulation of a real-time EEG stream, with LSL.

VR streams. It is important to note that each *StreamInlet* has no access to the XDF files. They only have access to their corresponding *StreamOutlet* buffers, where data is pushed at regular intervals. Two preprocessing steps are performed: clock synchronization and dejittering. It then pulls the available samples from the *StreamOutlet*, which are so far buffered (see Figure 12). We choose a sliding time window of 5s to be as close as possible to real-time processing, but spare computational expenses. This sliding window has a 4s overlap. Every 1s, it pulls the previous second of buffered data, and appends it to the 4s of data before. This way, we have a 5s time window of data.

Script This is how the toolbox script `realtime.py` looks like. It is based on functions of the toolbox and wraps `MNE-LSL` (see Code Fragment 2). Here, any function can replace `ANY_REALTIME_PROCESSING_FUNCTION(chunk)` to process chunks of data in realtime. For example, the whole pipeline of the toolbox, as described previously, can be here to mimic real-time processing. The size of the chunk depends on the frequency the algorithm pulls data. It can be as small as wanted, the pulling frequency just has to be increased accordingly.

```python
# Create the outlets - they have access to the XDF files
xdf_streams = load_xdf_streams("raw_eeg_data.xdf")
outlets = create_outlets(xdf_streams)

# Create inlets, they will automatically connect to the oulets
# They however have no access to the XDF files
inlets = create_stream_inlets()
open_streams(inlets)

# Start pushing XDF data to the outlet buffers
start_threads(outlets, ...)

# Pull chunks of data with inlets, connected to oulet buffers
while pull:
    chunk = pull_data(inlets, ...)
    ANY_REALTIME_PROCESSING_FUNCTION(chunk)

# Stop pushing data, close the outlets/inlets
stop_threads(threads)
close_streams(inlets, outlets)
```

Code Fragment 2: Real-time framework (`realtime.py`).

3.4. How can the pipeline be extended?

The current toolbox presents several opportunities for extension and refinement. Below, three primary directions for future development are outlined.

New Automatic Component Selection Method The three existing methods for automatic component selection exhibit limitations in robustness, particularly when applied to diverse or noisy datasets. To address this, future work could explore another approach. The idea of the third approach, to not rely on any baseline subject, looks like the best. We don't want anything subject-specific. We want something generalizable. However, the Hungarian algorithm is an optimization algorithm. It doesn't match the best matching components with each other. It ensures, in average, the matching cost is not too high. But we are just interested in the best matching components. If they match with each other across all subjects, it is enough, even if the other noisy components match very badly and produce a high cost.

It might be an interesting approach to then merge approach 3 with approach 2. Indeed, approach 2 first matches the best matching components with each other, and then the remaining ones... in the end, the noisy components match with each other. Having it in the approach 3 framework would allow us to select the good SPoC components without using a baseline subject.

Manual Component Selection While automation is preferred, manual selection remains a practical alternative, especially given that many subjects display one or two good-quality SPoC activation patterns. Developing a framework to manually

select the SPoC components, as for the independent component removal after ICA, could enhance the toolbox's usability. And even if it isn't automatic, it might be very fast to detect among the SPoC activation patterns which one(s) is(are) interesting. It might improve the results a lot, and remove a lot of noise.

Wrapping the Realtime Script The current implementation of the `realtime.py` script requires users to write their data processing and analysis in a template script, as described in Code Fragment 2. It however limits usability. To improve this, we can design a decorator or wrapper around the script, enabling users to define their processing functions externally. This approach would promote cleaner code and facilitate the integration of custom logic. The real-time framework would be easily used like shown on Code Fragment 3.

```
from toolbox.realtime import realtime

@realtime("raw_eeg_data.xdf")
def ANY_REALTIME_PROCESSING_FUNCTION(chunk):
    # Do whatever you want with the chunk of data
    pass

ANY_REALTIME_PROCESSING_FUNCTION()
```

Code Fragment 3: Using the real-time framework as a decorator.

4. Validation

4.1. Experiment

4.1.1. Use case background

Personal Space (PS) has been defined in 1978 by *Hayduk et al.* [20] as *"the area individual humans actively maintain around themselves into which others cannot intrude without arousing discomfort"*. This study also defined the stop-distance procedure we will later use, to determine the PS of an individual. The PS is specific to each individual, and many factors can influence it: facial expressions [47], density of people and cultural differences [45], age and the gender of the individuals [30][22].

To investigate PS, **virtual reality** (VR) is a powerful tool. It offers researchers precise experimental control while preserving the authenticity of social interactions [47][55]. The immersive nature of VR accurately replicates the behavioral dynamics of personal space observed in real-world settings, including the PS distance. In *Hofmann et al., 2021* [21], the researchers use VR environments to study emotional arousal. They have shown that this is to be observed in the parieto-occipital region of the brain.

For the final K3VR training, one would need the VR environment to communicate with the brain signals of the user. This is called a **brain-computer interface**. It has been widely studied in the last decades. It has already been used on single-trial EEG [5], for example. BCIs allow to decode brain signals in real-time, and use them to control external devices. The Berlin Brain-Computer Interface is one example of such advanced BCI system [6][38][7]. One problem that arises with BCI is the long calibration time required, on each new subject. Machine learning methods however reduce this time, without significant performance loss [14].

To use such BCI in real-time, one can use **Lab Streaming Layer** (LSL). It is a system for the unified collection of time series data in research experiments [25]. The `Python` package `MNE-LSL` is used for the real-time framework of the toolbox [50].

4.1.2. Experimental design

The experiment aims to validate the toolbox presented in **Chapter 3** by decoding the distance between a subject and an avatar in a virtual reality (VR) environment, using EEG signals. We also aim to detect personal space (PS) intrusion. The hypothesis is that the EEG signals vary based on the distance of the avatar,

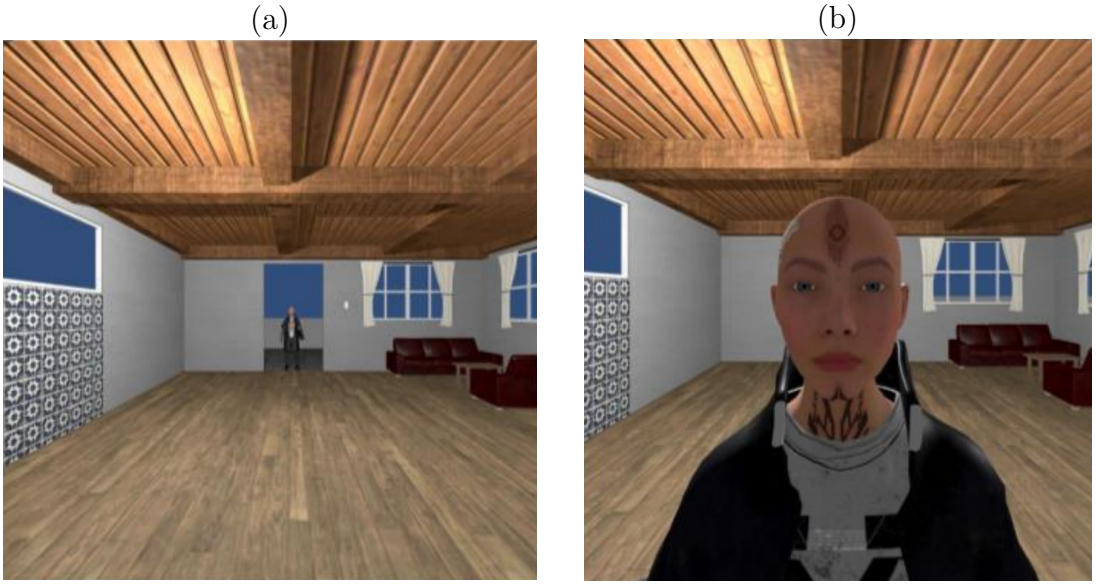


Figure 13: Avatar distance: (a) PS respect vs. (b) PS intrusion.

Figure 14: Avatar's facial expression: (a) neutral face vs. (b) angry face.

particularly when it intrudes into the subject's PS, and that these variations can be decoded using the toolbox.

The subjects wear a VR headset where the avatar is shown, and an EEG cap to record their neurophysiological signals. The experiment employs a 2×2 within-subjects design to investigate the effects of **avatar distance** (see Figure 13) and **facial expression** (see Figure 14) on neurophysiological responses. The factors and their levels are summarized in Table 1.

Factor	**Levels**
Avatar Distance	PS intrusion
	PS respect
Facial Expression	Angry
	Neutral

Table 1: Experimental factors and levels.

Participants A total of 20 subjects (10 females, 10 males) were recruited for the study. They were standing during the experiment.

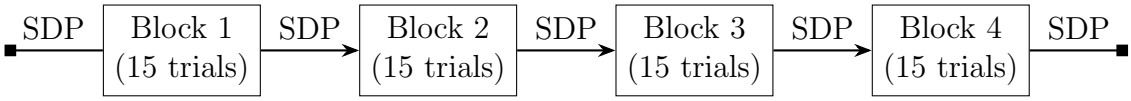


Figure 15: Experimental design. The overall 60 trials are shuffled before being separated into 4 blocks.

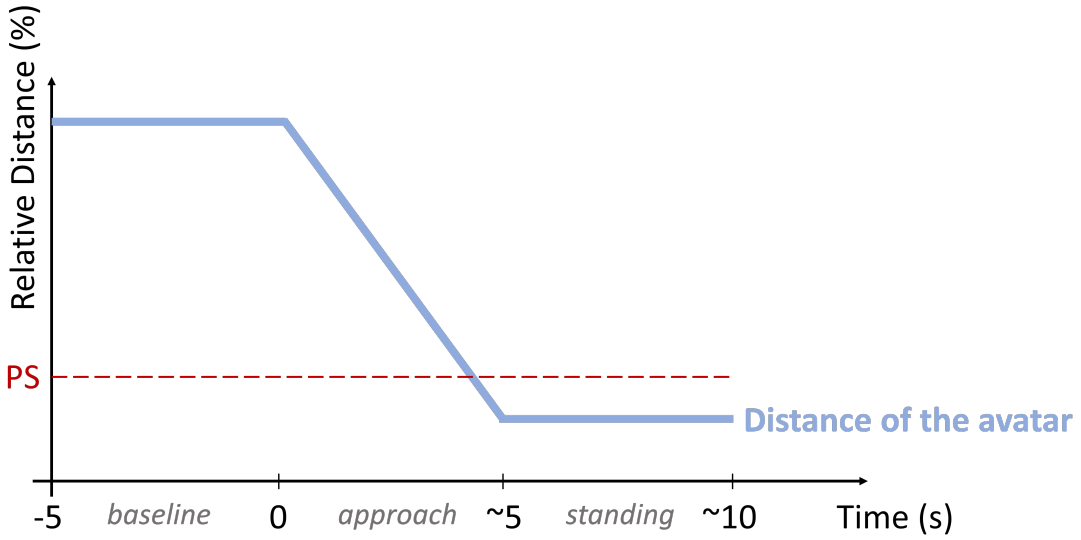


Figure 16: Procedure for one trial.

Procedure Each participant experienced four experimental conditions, resulting from the combination of the two factors. Each condition was presented across 15 trials, totaling 60 trials. The shuffled trials were organized into four blocks (see Figure 15), with breaks between blocks. In each trial, the following sequence occurred (see Figure 16):

- Baseline: the avatar stood 8 meters away from the participant for 5 seconds (baseline).
- Approach: the avatar then walked toward the participant and stopped at a predetermined distance, either intruding into or respecting the participant's personal space, with either a neutral or an angry face.
- Standing: the avatar remained at the stopping distance for an additional 5 seconds.

The PS of each participant is determined at the beginning of the experiment, between each block, and at the end (see Figure 15). It is done via a stop-distance procedure (SDP) [20]: the avatar comes closer to the participant, and the latter says the moment when he starts feeling uncomfortable. The distance at which the avatar is located at that moment is defined as the PS of our subject.

Stimuli A gender-neutral, human-like avatar was displayed in the VR environment. The avatar's stopping distance (PS respect or intrusion, see Figure 13) and facial expression (neutral or angry, see Figure 14) were manipulated according to the experimental conditions.

4.1.3. Data collection

EEG recording Neurophysiological data were recorded using a 64-channel EEG system (LiveAmp, BrainProducts) at a sampling rate of 500Hz. The EEG cap was wireless to allow for natural movement within the VR environment.

Avatar distance recording The distance between the participant and the avatar was continuously recorded throughout each trial to correlate with EEG responses, thanks to the VR headset (Meta Quest Pro).

4.2. Use of the toolbox

The goal of this case study, in line with the title of this Master's thesis, is to decode the proximity of the avatar in the virtual environment, from the EEG signals of the subject. To do that, the toolbox presented in **Chapter 3** is used, and its pipeline leveraged.

It applies the preprocessing steps described. For the feature extraction, SSD is used on the alpha frequency range (8-13 Hz) [3]. SPoC is then used, with the behavioral target variable **z** being the ground truth distance of the avatar to the subject, over time.

4.3. Results

4.3.1. Interpolation

We wanted to do an interpolation per $5s$ chunk of data. A grid search was computed to determine two hyperparameters of the interpolation: the interpolation method (linear, nearest, quadratic, cubic, etc.) and the number of samples to look back in the previous chunk (0 to 14). This was done by 5-fold cross-validation (CV). The CV-score for each combination of hyperparameters was computed, on all data of all subjects (Figure 17).

We can observe that the number of samples to look back doesn't have a strong influence on the CV-score. It can be guessed that the interpolation is mainly determined by the points of the sample. The previous points have a minor influence on the interpolation curve, thus on the CV-score. A minimum is found for **5 samples** to look back: this value is kept.

Concerning the interpolation method itself, the **quadratic** method has the lowest mean and standard deviation. It is thus the one kept.

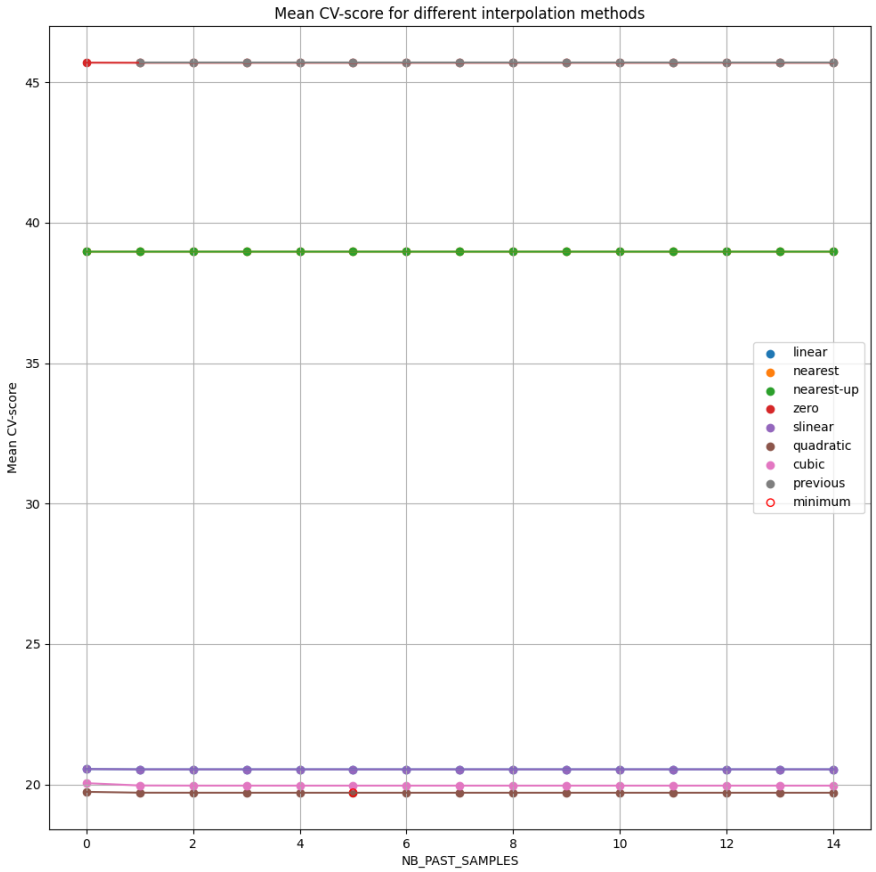


Figure 17: Cross-validation scores. Grid search on interpolation methods + number of samples to look back.

It can be seen how the final interpolation looks like on a snippet of the data (Figure 18). The blue line is the original signal, the orange line is the interpolated signal. Both have a 500Hz sampling frequency. The interpolation looks good, and follows the original signal well. It has however now an exact spacing of $2ms$ between each timestamp.

4.3.2. Results of the bad channel removal

The power spectrum allows for quick identification of bad channels, as a first approach. This is for example the case for subject 45 for electrode *FC2* (Figure 19), which has a much higher power than the other channels.

We could, in a second approach, use the maximum activity heatmaps (see Figure 20). For subject 45, the necessity to remove *FC2* is confirmed.

Finally, the LOF scores allow for a double-check, and even to discover unidentified outliers. In Table 2, it can also be seen that *FC2* is much different than the other channels. In addition, the power spectrum and maximum activation heatmap for channels *Pz, Fp1, AF7, F8* can be checked. They may be abnormal according to their LOF scores above threshold. After this check, they are kept,

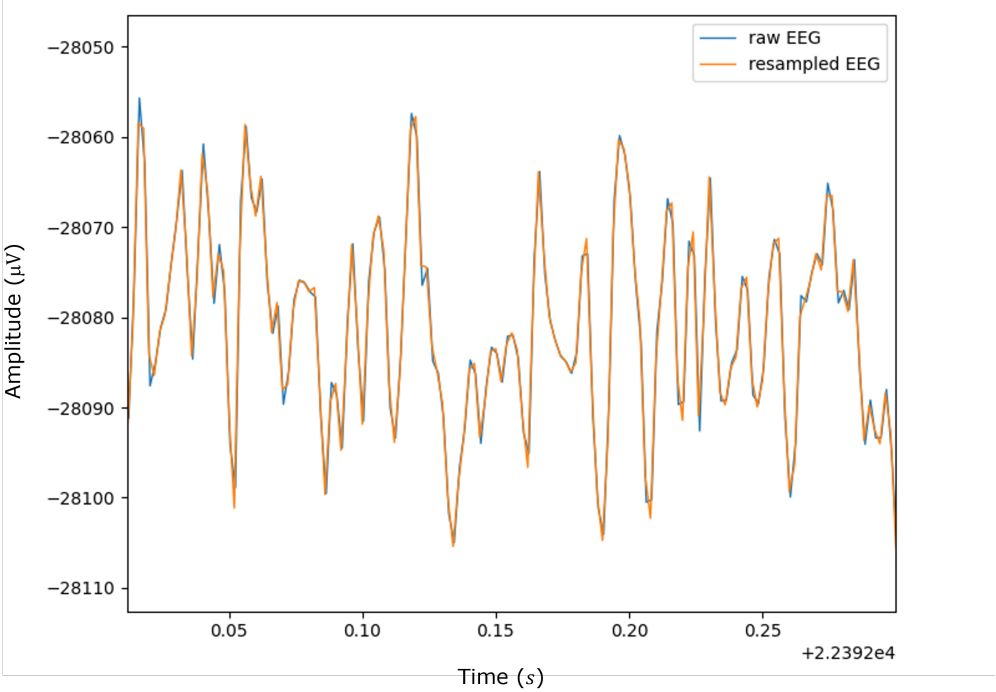


Figure 18: Final interpolation.

EEG channel	LOF Score
FC2	49.28
Pz	6.85
Fp1	2.43
AF7	2.24
F8	2.23

Table 2: LOF scores over 1.4 - the outlier threshold - for subject 45.

since they don't present anything unusual.

On average, 0.63 ± 0.86 channels were removed per subject.

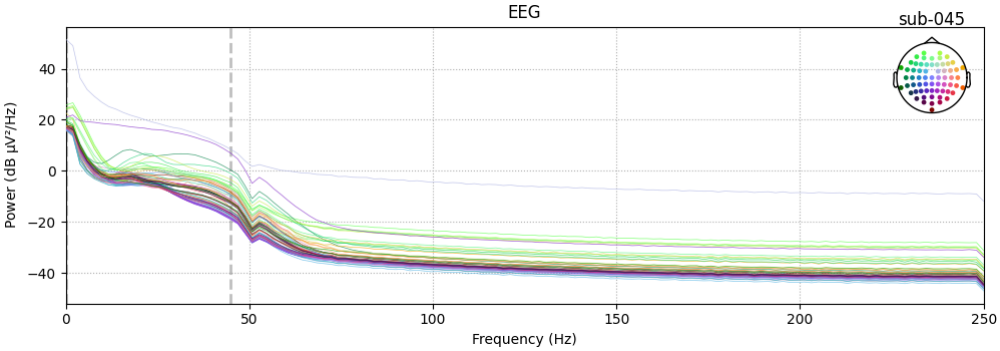


Figure 19: Power spectrum (subject 45).

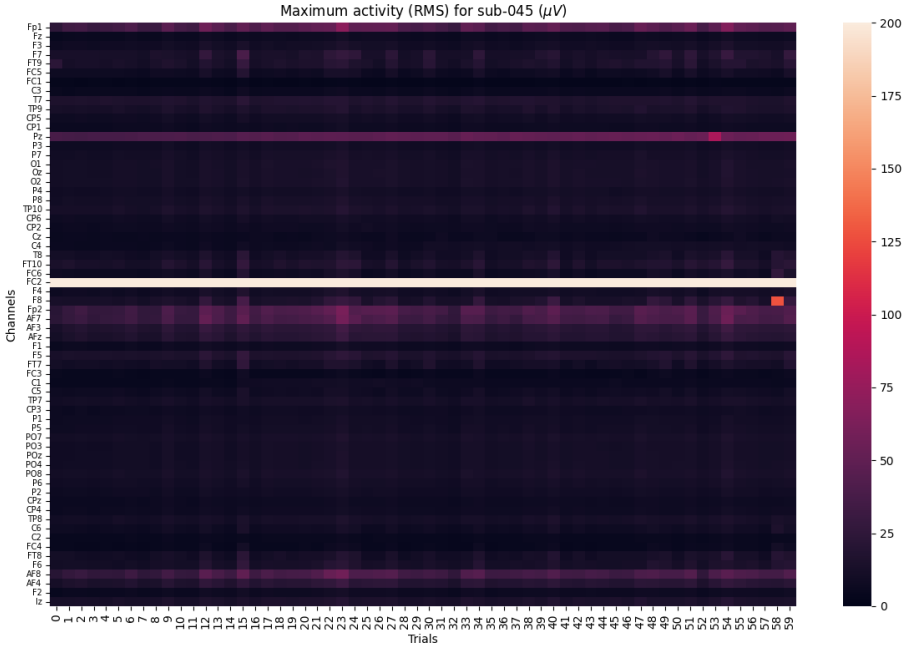


Figure 20: Maximum activity heatmap (subject 45).

4.3.3. Result of the ICA

After computing the ICA, artifacts identified were removed by subtracting the signal corresponding to them from the global signal.

This is what 20 Independent components (ICs) look like, for two different subjects (Figure 21).

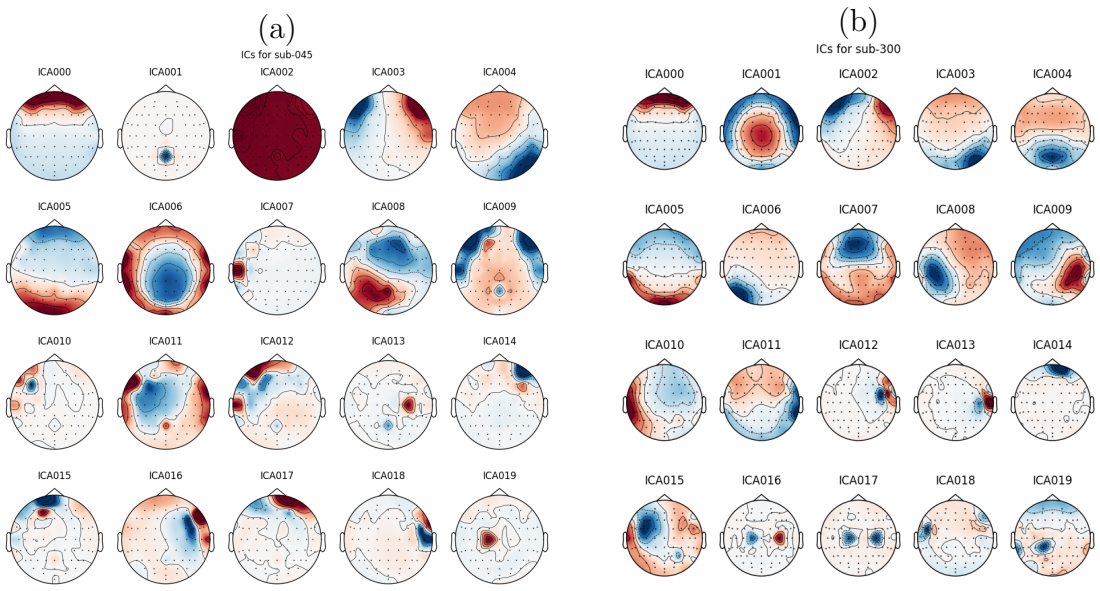


Figure 21: 20 Independent components found by the ICA, for (a) subject 045 and (b) subject 300.

Eye artifacts For instance, we can clearly see the eye artifacts. Eye movements correspond to IC 3 for subject 045 and IC 2 for subject 300. Eye blinking corresponds to IC 0 for subject 045 and subject 300. On Figure 22, we can see the power spectrum of these two ICs. They here have a characteristic profile of an eye blink artifact, with a peak in the low frequencies and the spectrum then decreasing substantially.

Muscle artifacts We can identify muscle artifacts on IC 12 and 13 of subject 300 (Figure 23). On the topography, they are located on the outside of the scalp. Moreover, they have a power spectrum with stronger power in higher frequencies (> 20Hz). This is characteristic of a muscle artifact.

Other artifacts The last type of IC to remove is the channel noise, that may be caused by a bad contact between the electrode and the scalp. We have an example of channel noise on IC 1 of subject 045 (Figure 24). It is located very precisely on a given electrode, and it shows a very flat broadband on the power spectrum. As explained on the practicing website from *Pion-Tonachini et al., 2019* [44], this is characteristic of channel noise.

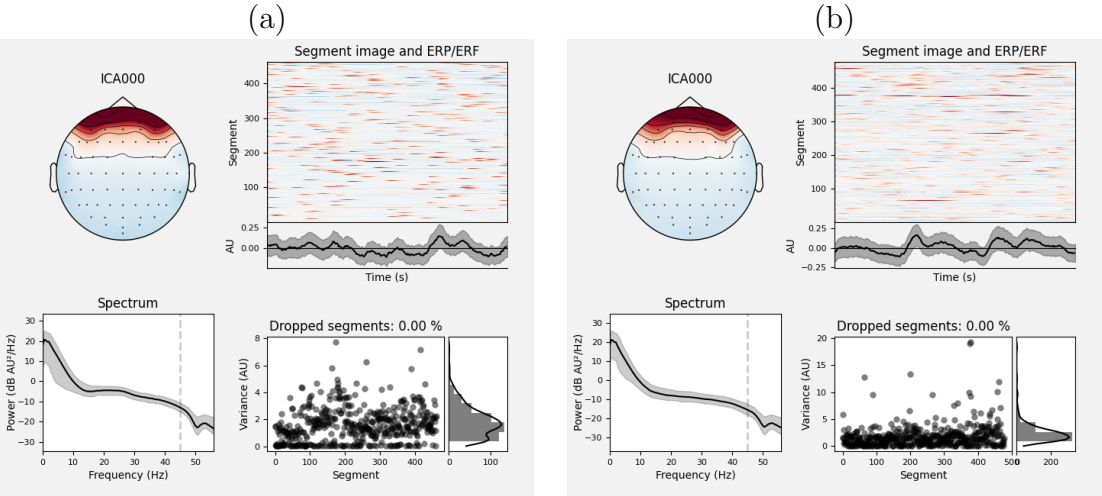


Figure 22: Eye blink - (a) independent component 0 for subject 045 and (b) independent component 0 for subject 300.

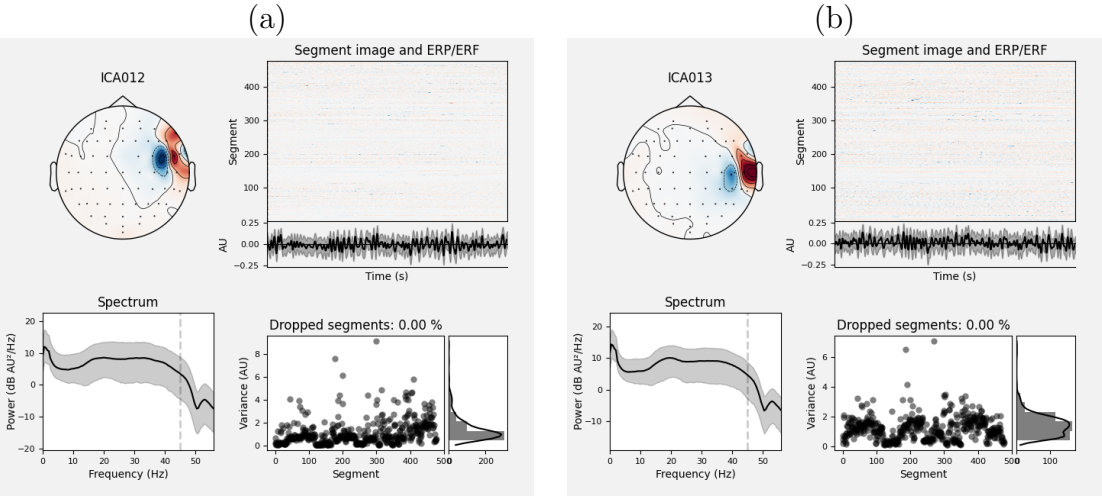


Figure 23: Muscle artifact for subject 300 - (a) independent component 12 and (b) independent component 13.

In the pipeline developed, the indices of the ICs to remove were saved in a JSON file, for each subject. The type of artifact was also saved. This way, the same artifact removal can be easily reproduced, and it is easy to inspect what was removed, even later.

4.3.4. SSD Results

The spectral ratios and elbow point are shown on Figure 25. In average, 10.69±1.04 components were kept for each subject.

On Figure 26, we can see SSD patterns for subject 300. These are the zones of the scalp where alpha oscillations are mostly emitted. On pattern 5, we can see alpha frequencies in the parieto-occipital zone, where we would expect emotional

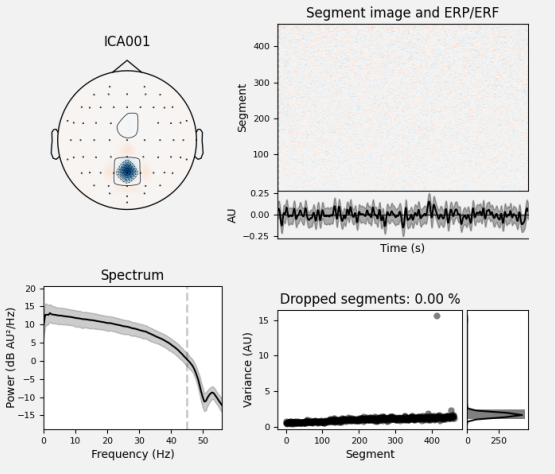


Figure 24: Channel noise - independent component 1 for subject 045.

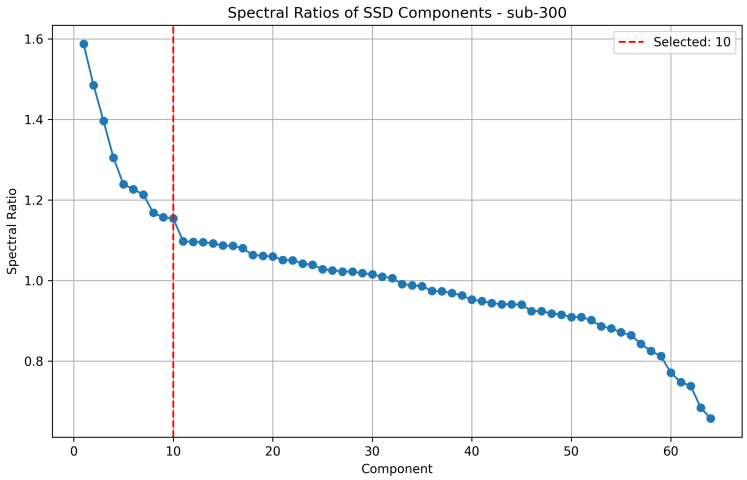


Figure 25: SSD spectral ratios for subject 300. The red dashed line corresponds to the elbow point.

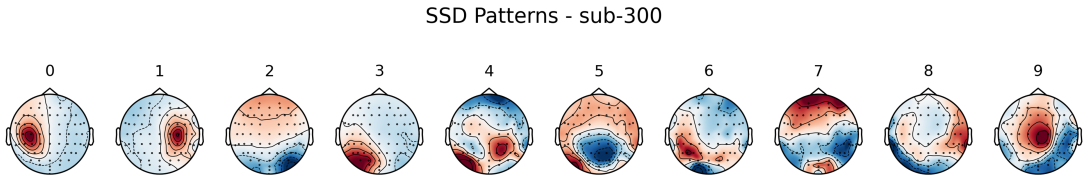


Figure 26: The ten selected SSD patterns for subject 300. It has been sorted by the absolute value of the eigenvalues λ, which corresponds to the SNR (resp. 1.60, 1.45, 1.40, 1.35, 1.30, 1.28, 1.25, 1.22, 1.20, 1.18).

arousal [21]. We can also see many other zones of the scalp where alpha oscillations

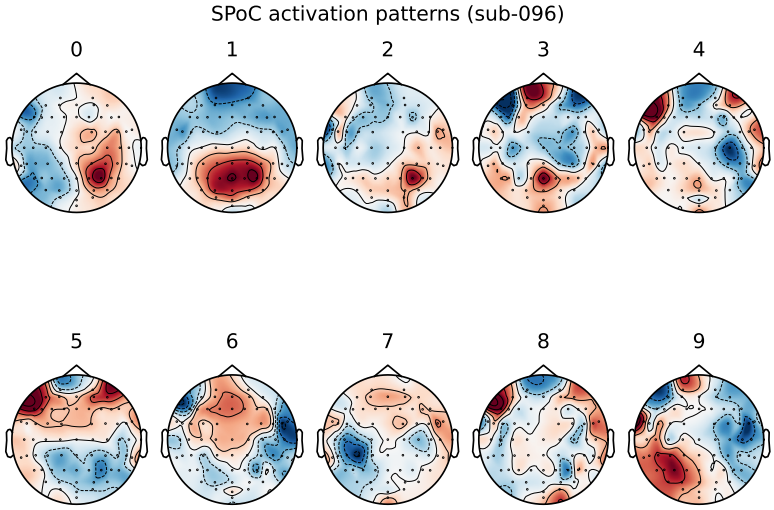


Figure 27: SPoC activation patterns for subject 96.

are emitted. They might be noise, or other brain signals, like pattern 7. There is still processing to do in order to isolate the signal of interest.

4.3.5. SPoC Results

We can see on Figure 27 the result of SPoC for subject 096. Component 1 is a very interesting pattern, in the zone where brain oscillations are expected [21]. There is however also a lot of noise (components 3 to 9), and it is hard to interpret

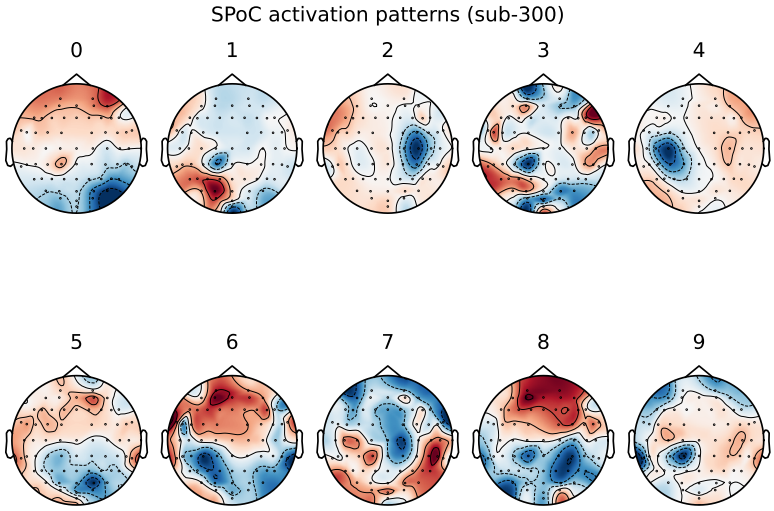


Figure 28: SPoC activation patterns for subject 300.

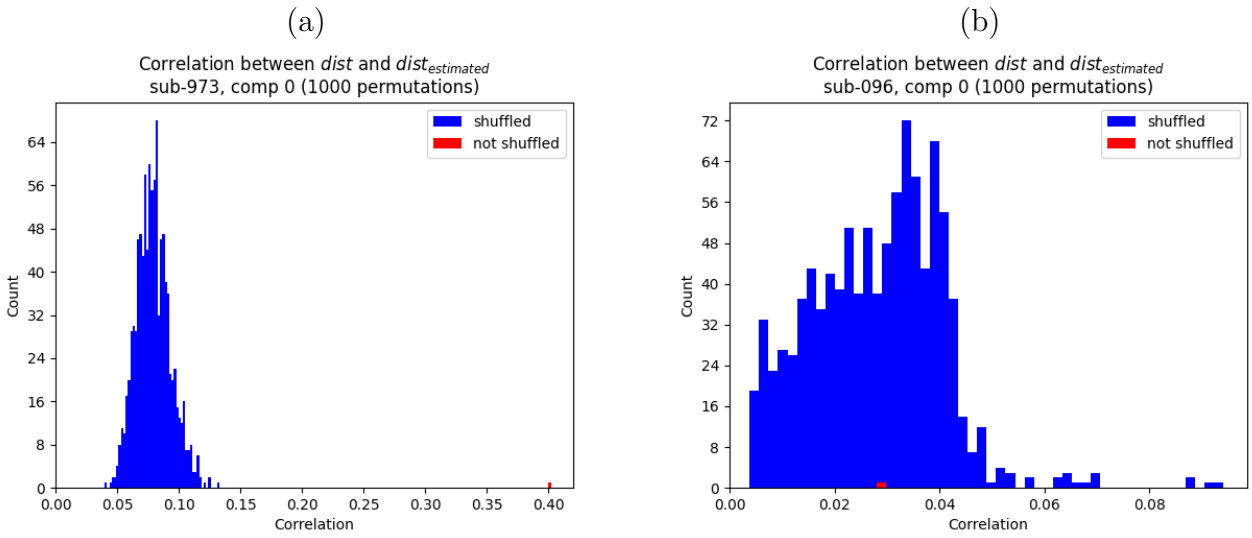


Figure 29: Component 0 after permutation analysis ($dist$ and $dist_{estimated}$ respectively correspond to z and $\tilde{z}/\tilde{z}_{\text{shuffled}}$). (a) Significant component (subject 973), (b) insignificant component (subject 96).

components 0 and 2. One first challenge will be to automatically select the good SPoC components. They have been sorted by their eigenvalue $|\lambda|$. It is however not reliable enough to identify the interesting or the noisy components.

Second observation, some subjects don't exhibit good SPoC components at all, as we can see for subject 300 in Figure 28.

Many components look "patchy", and don't really look like brain activity. It might be due to bad data quality for this subject, and to the fact that the EEG cap was wireless. This allows more interference than a wired cap. It might also be due to the preprocessing. In particular during the ICA, some brain activity might have been removed by mistake, or too much noise kept. The ICA was done twice, since it was already identified as probably hindering the analysis. It was however very time-consuming and didn't improve the results much. This point would be investigated more with more time. More activation patterns for other subjects are shown in the Appendix B.

4.3.6. Results of the permutation analysis

The permutation analysis allowed for identification of which components were significantly correlated with the target variable $\mathbf{z}$ (distance to the avatar), and which ones relied on spurious correlations.

Homogeneous subjects On Figure 29 (a), we can see the distribution of correlations between $\mathbf{z}$ and the 1000 predictions $\tilde{\mathbf{z}}_{\text{shuffled}}$ on component 0 of subject 973, after permuting $\mathbf{z}$ 1000 times. These are in blue. It is compared to $\tilde{\mathbf{z}}$, the baseline prediction without shuffling $\mathbf{z}$, in red. We can see that the correlation of the real prediction is much higher than the correlations of the predictions with permuted

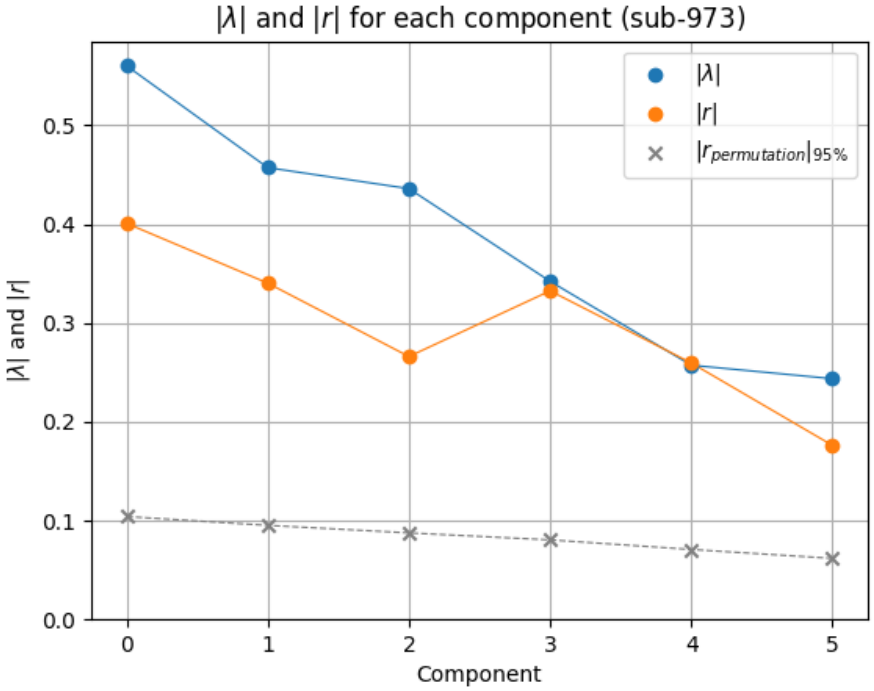


Figure 30: Eigenvalues $|\lambda|$ in blue and correlations $|r|$ in orange, between ground truth distance z and estimated distance $\tilde{z}$, for the first 5 components of subject 973 (sorted by the absolute value of their eigenvalue). The grey dashed line corresponds to the threshold above 95% of the correlations between z and estimations with permutation $\tilde{z}_{\text{shuffled}}$.

$\mathbf{z}$. This means component 0 doesn't rely on spurious correlations for subject 973. It is significant.

On the contrary, it can be seen on Figure 29 (b) that component 0 for subject 96 isn't significant. The correlation of the real prediction $\tilde{\mathbf{z}}$ is indeed not higher than 95% of the correlations of the predictions $\tilde{\mathbf{z}}_{\text{shuffled}}$. This means this component relies on spurious correlations. It has not been kept for further analysis.

On Figures 30 and 31, we can see these observations for the first 6 components of a subject. On the first one, all components have a correlation significantly higher than the threshold (95% of the permuted correlations are under this threshold). We can also see to what extend it is linked to their associated eigenvalue λ, in absolute value. On the second one, we can see a subject where none of the components is significant. All correlations are under the threshold. This means all components rely on spurious correlations, and we did not keep any of them for further analysis.

Heterogeneous subjects For many subjects though, this was more complicated. It was not either all components significant, nor all components insignificant. Most subjects had a mix of both. In addition, some components were very close to the threshold, and it was hard to decide if they were significant or not. An additional

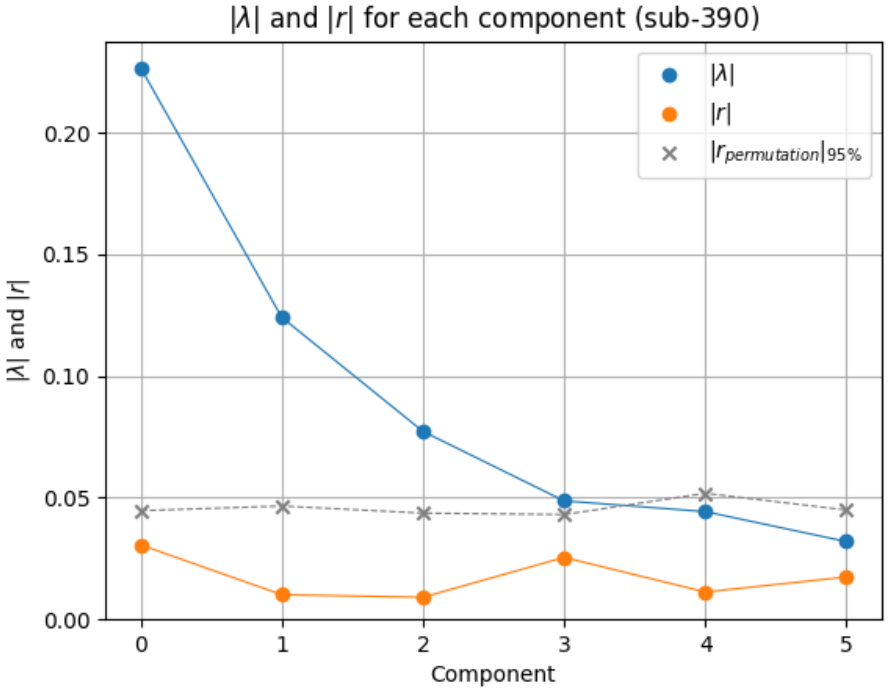


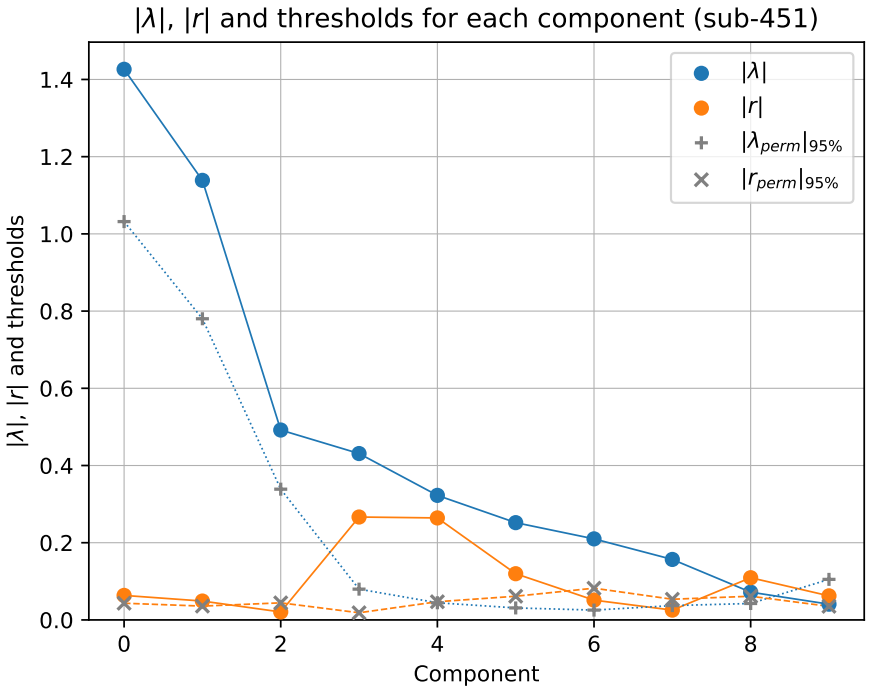
Figure 31: Eigenvalues $|\lambda|$ in blue and correlations $|r|$ in orange, between ground truth distance z and estimated distance $\tilde{z}$, for the first 5 components of subject 390 (sorted by the absolute value of their eigenvalue). The grey dashed line corresponds to the threshold above 95% of the correlations between z and estimations with permutation $\tilde{z}_{\text{shuffled}}$.

threshold is plotted (see Figure 32): $|\lambda_{perm}|_{95\%}$, which is the value below which 95% of the eigenvalues obtained after fitting SPoC on the data with permuted z are found. It was an additional information to help decide if a component was significant or not. If its eigenvalue is under this threshold, it is also considered as insignificant.

On Figure 32 (a), only components 3 and 4 are really above both thresholds. Component 3 looks like activity from the parieto-occipital zone, where we expect emotional arousal (see Appendix B). Component 4 however shows patchy activity in many zones, it is hard to interpret if it is a real brain signal: it might be noise. On Figure 32 (b), only components 1, 4 and 9 are insignificant and were excluded. The interesting components are 5 and 6. They were kept, but we need to find a way to identify and exclude the others, which are significant but still not carrying the signal we are interested in.

Eigenvalue $|\lambda|$, a good way to select components? We could compare the eigenvalues $|\lambda|$ with the correlations $|r|$ between $\mathbf{z}$ and $\tilde{\mathbf{z}}$. On Figure 30, $|\lambda|$ gives quite a good idea of the correlation $|r|$. At least the component with the highest eigenvalue has also the highest correlation. However on Figure 31, it is not

(a)



(b)

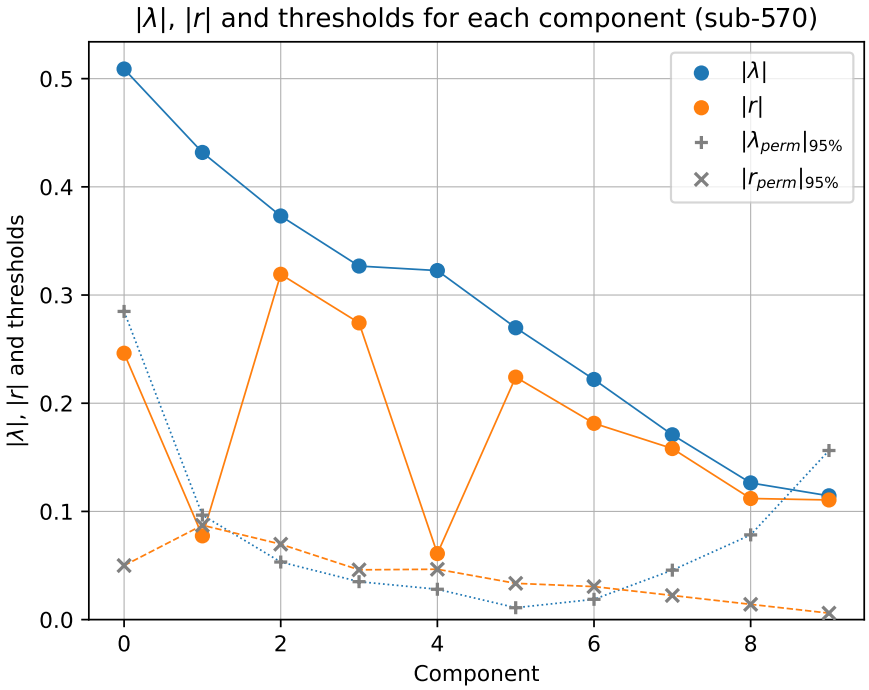


Figure 32: Eigenvalues $|\lambda|$ in blue, correlations $|r|$ in orange, and thresholds with dashed/dotted lines. Components are sorted by the absolute value of their eigenvalue. (a) subject 451, (b) subject 570.

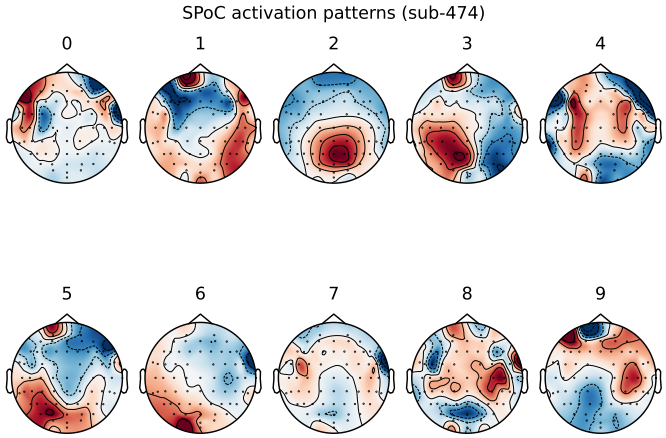


Figure 33: Activation patterns for subject 474.

the case. Reordering the components by their correlations would give a totally different order. In addition, on Figure 32, the second component in $|\lambda|$ is even insignificant. The eigenvalues aren't reliable enough to order the components by order of relevance and significance.

To give an intuition on the decorrelation between eigenvalue and correlation, we can say the following:

1. The higher the eigenvalue, the more the component comodulates with the target variable $\mathbf{z}$. The correlation is however obtained from distance prediction, which is a separate step to eigenvalue decomposition. It looks like a good comodulation helps to have a good prediction, but it doesn't always have to be the case.

2. Highest eigenvalues aren't always linked with the most significant components. This is because the eigenvalue is a measure of the strength of the relationship between the component and the target variable, but it does not account for noise or spurious correlations. Therefore, a component with a high eigenvalue may still be insignificant.

We thus needed to find another way to select the components to keep to perform the final distance estimation.

4.3.7. Results of SPoC component selection

After excluding the insignificant and spurious correlated activation patterns, some of the remaining ones still did not look like what would be expected.

Approach 1 (single-subject): Select prototypes

The γ method gave the best results, even better results than with the δ method. Different values of k have been tried, and the best results are with $k = 3$.

Component	Score
0	0.0902
1	0.0171
2	**0.6478**
3	0.3952
4	0.2702
5	**0.5812**
6	**0.5438**
7	0.4290
8	0.2288
9	0.1064

Table 3: γ scores for the SPoC components of subject 474. The best 3 scores are in bold.

On Figure 33, the most interesting pattern is component 2, as it exhibits clear activations in the parieto-occipital region. In Table 3, component 2 has the highest score and was thus selected. The second best score is component 5. It shows some activity in the parieto-occipital region—which might be interesting—but also artifact on the prefrontal cortex—it is thus not sure. The third best is component 6. It mostly looks like channel artifact and is thus not interesting. The threshold would then be something around 0.56.

However, this method was not reliable enough. For some subjects, it was very successful, but for some others, the best activation patterns were not selected. It was also hard to find a consistent threshold score across all subjects. In other examples, a threshold of 0.56 would select either no component or too many components. Finally, the patterns were not consistent across subjects: the best score pattern from one subject could be quite different from another. This method does not provide pattern alignment.

Approach 2 (across-subjects): Align over one baseline subject

This was not the optimal approach since it required one subject as a baseline. No subject-specific approach is desired. In addition, the results were quite similar to the next approach, which was more generalizable.

Results for this approach are to be found in Appendix C.

Approach 3 (across-subjects): Align without baseline subject

Results of the alignment are to be seen on Figures 34 and 35.

The reordering looks completely different than when it was ordered by $|\lambda|$. Appendix D shows how each original component index matches the new global components scheme, for each subject. For the sake of simplicity, only the indexes are presented there.

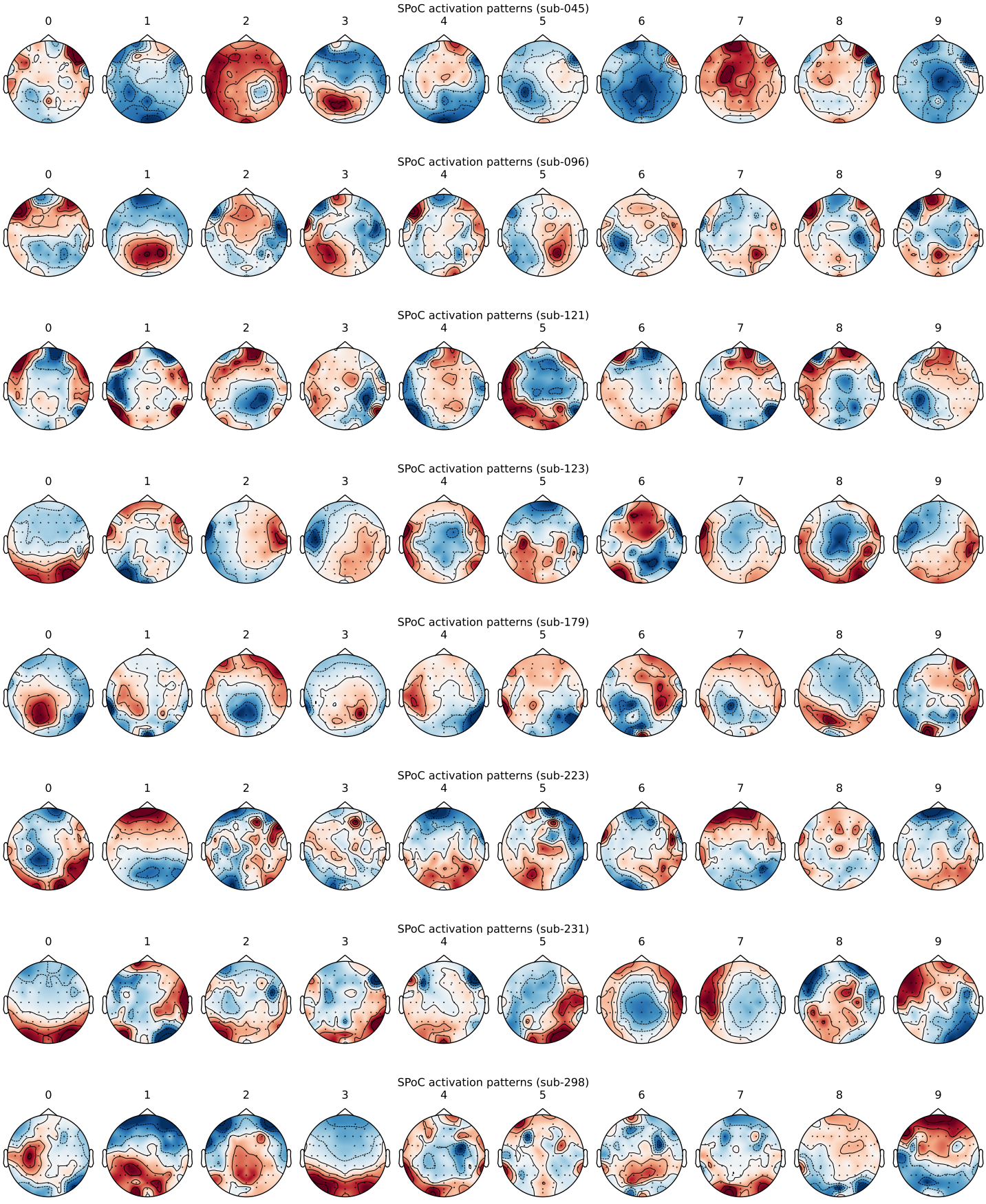


Figure 34: Aligned patterns on global patterns, for each subject (1).

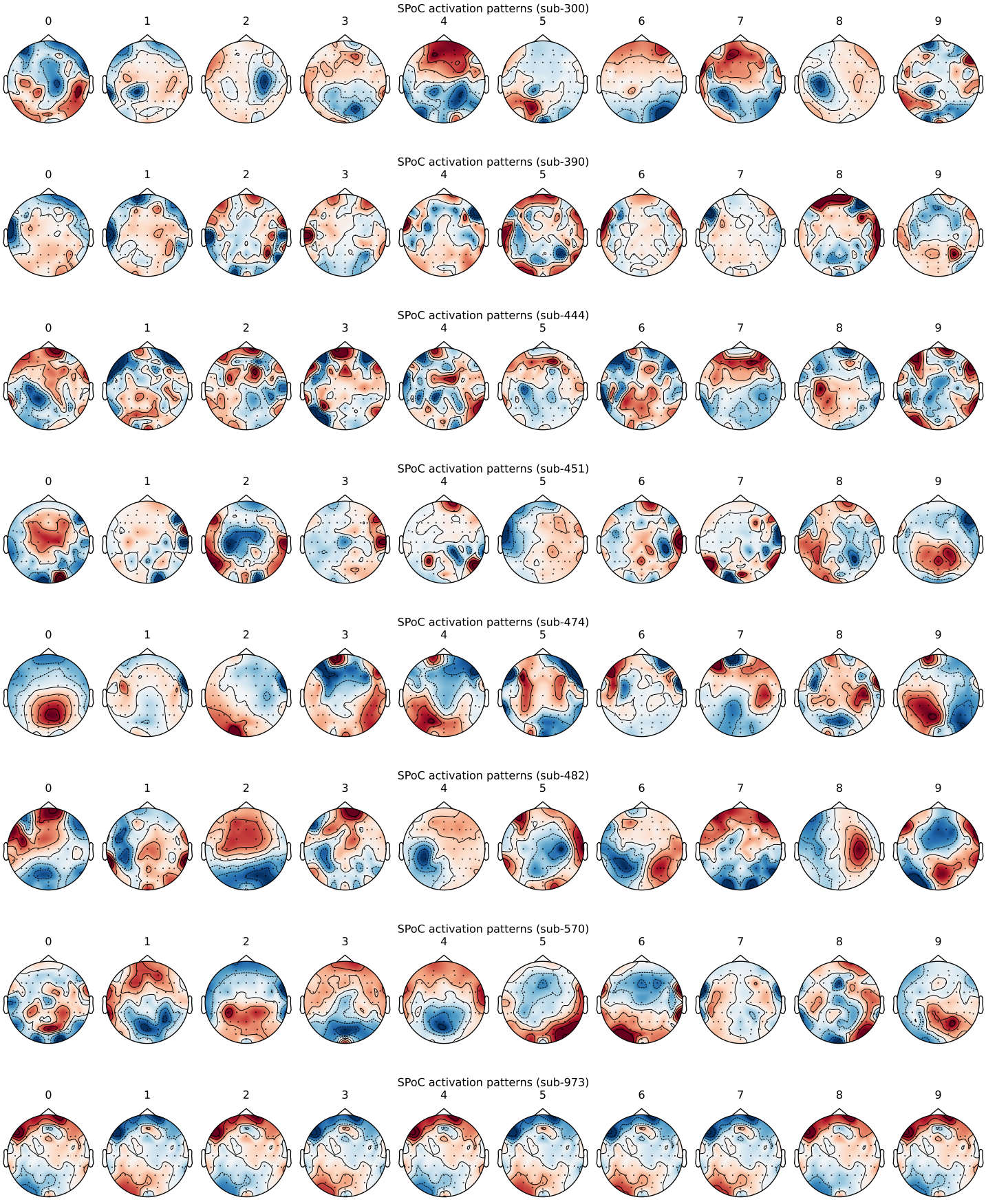


Figure 35: Aligned patterns on global patterns, for each subject (2).

The results are hard to interpret and the alignment unfortunately didn't really work. Very similar patterns across each column, or at least across certain columns, would be expected. This is not the case here. More is elaborated in the **Chapter 5 - Discussion**, but here is already a few explanations on why it might not have worked:

1. The data quality might not be good enough. The wireless EEG cap might have created too much noise and interferences.

2. The preprocessing might not be optimal. The ICA in particular might have removed some brain activity by mistake, or have kept too much noise.

3. The alignment method might not be optimal. As explained in **New Automatic Component Selection Method 3.4**, a mix between approaches 2 and 3 might be more efficient. The Hungarian produced the "less bad" alignment given all components, including the noisy ones. Focusing first only on the best ones, like in approach 2, might have yielded better results. We should however keep the advantage of approach 3, which is not needing a baseline subject.

4. Some subjects should maybe be entirely excluded, like subjects 390 or 444. They had no good-looking SPoC pattern at all. They were however taken into account in the Hungarian algorithm, and might thus hinder the alignment.

4.3.8. Results of the distance regression

Figure 36 is the result of the distance prediction for subject 179 (cropped on around 4 minutes). It used the first component, the one which best comodulates with $|\lambda|$. It exhibits activity in the parieto-occipital region. The signal had also been smoothed, as a post-processing step. The associated correlations r are shown.

Two things to note. The toolbox is good to predict the distance, even if there is still room for improvement. It can detect closeness and farness to the avatar. After smoothing, it reached a correlation of 0.45 with the ground truth avatar distance, for the whole experiment on this subject. It is a significant improvement, as compared to the correlation between predicted emotional arousal and ground truth of *Hofmann et al., 2021*—0.25.

However, the prediction is lagging behind the real distance. It also always stops around 200% (of the PS) when the avatar is close. This prevents discriminating PS intrusion from PS respect. It is then impossible to predict PS intrusion and use the function `detect_ps_intrusion()`. This is the case for all subjects. The SPoC component doesn't manage to capture the brain oscillations linked with PS intrusion. This might be because SPoC hasn't managed to capture it, or because this information is in another component. Many components have been tried for the regression, also an average of multiple ones and it never worked. However, a more systematic approach to test all component combinations (kind of a grid search) could have helped. Otherwise, a manual selection of the good-looking component(s) might have helped.

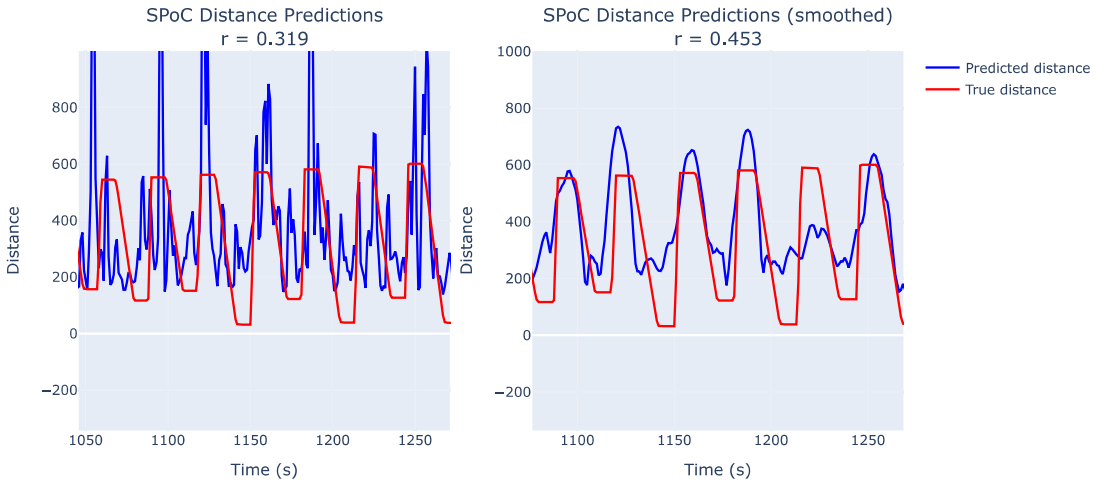


Figure 36: Distance prediction for subject 179, using component 0, on around 4 minutes. In red is the ground truth avatar distance, in blue is the prediction. On the left, this is the raw signal. On the right, it has been smoothed.

4.3.9. Results of the detection of events

We can see on Figure 37 the results of the close/far phase detection for subject 179, on the same time window as the distance regression shown just before. It can be seen that despite some failures (in orange), most of the close/far phases have been correctly detected (in green).

The overall accuracy over all subjects is $47.12\% \pm 15.38$. It is significantly higher than a dummy detector, which would have an accuracy of 33.33%. It should be noted that it varies a lot across subjects. This depends a lot on the SPoC component selection, and if the right component had been kept. Here, the component with highest comodulation with the avatar distance was kept, since the automatic component selection was not consistent enough. But here, a manual selection of the best-looking component(s) might have improved the results. This would be an idea for future work.

On Figure 38, the results for subject 96 can be seen, on the whole experiment. Here the accuracy is 66.67%. It means, $\frac{2}{3}$ of the close and far standing phases have been correctly detected, which is very satisfying. Results above 66% have been obtained for 3 other subjects: 300, 570 and 973.

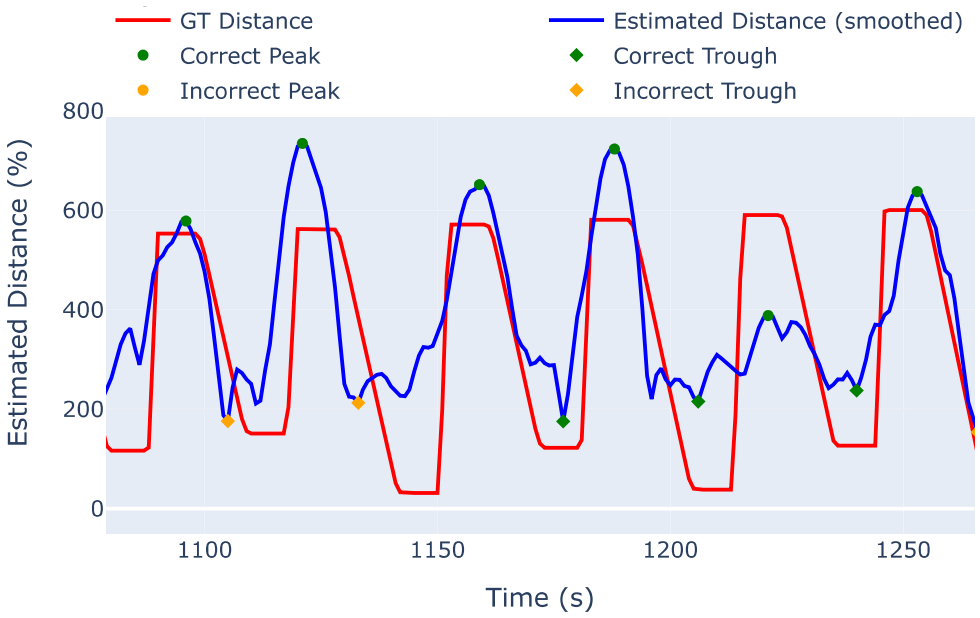


Figure 37: Close/far phase detection for subject 179, using component 0, on around 4 minutes. In green are the correctly detected phases, in orange the failures.

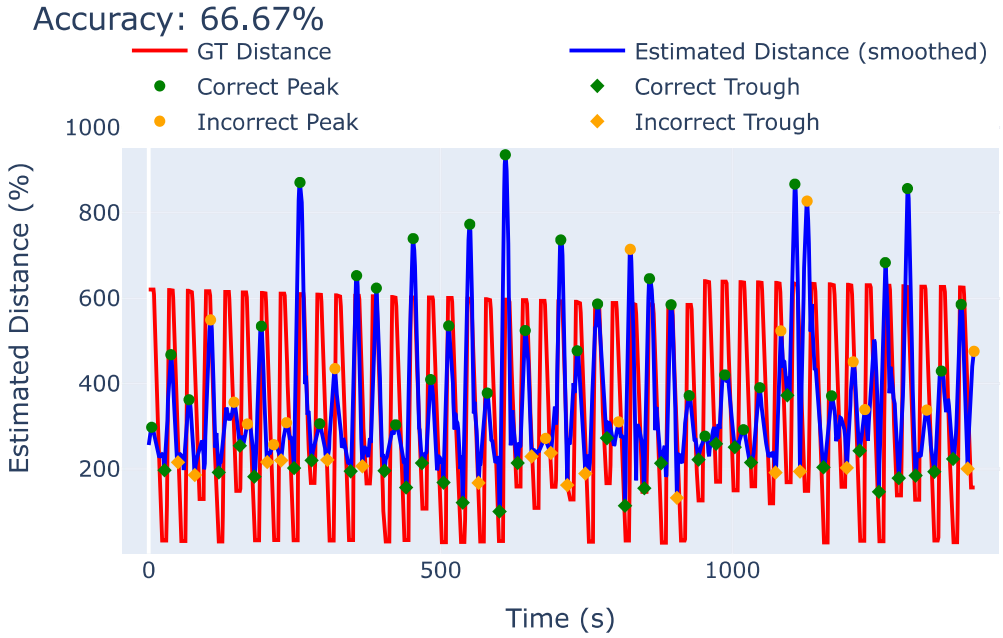


Figure 38: Close/far phase detection for subject 96, on the whole experiment. In green are the correctly detected phases, in orange the failures.

5. Discussion

This thesis aimed to develop a semi-automated, modular, and real-time-ready pipeline for decoding neurophysiological signals from EEG data, with a specific focus on assessing proximity in a virtual environment. By wrapping existing methods and implementing missing ones, we created a robust toolbox that successfully processed raw EEG data, extracted relevant features, and regressed a distance. This pipeline runs in a few minutes per subject, completing the whole workflow from A to Z, with minimal user input. Below, we discuss the interpretation of these results, their limitations, and broader implications for EEG-based applications.

Interpretation of Results - The Toolbox The application of the pipeline on this case study provided valuable insights into the effectiveness of the developed toolbox. Below, we discuss the key findings and their implications. The toolbox demonstrated significant **advantages in usability**. The analysis pipeline executes all steps sequentially, ensuring a smooth and effective workflow. Throughout the process, clear and consistent logs provide real-time updates to the user. The intuitive design of the toolbox guides users effortlessly through each stage of the process, making the toolbox accessible even without a deep understanding of the code. In terms of efficiency, the pipeline completed a single-subject analysis in a few minutes, which is satisfying. The **automation** works well. The proposed pipeline operates in a semi-automated manner: while manual intervention is required for the identification and removal of bad channels and artifacts, all subsequent steps are fully automated. It is also fully **modular**. Before each step, the toolbox loads the previous results and, after each step, it saves the current results. This allows full modularity of the pipeline, given that all steps are independent of each other (they however must be run in the correct order, of course). It is then possible to run only parts of the pipeline by commenting out undesired functions. Many combinations of steps were tested and the pipeline behaved robustly without crashing. Finally, the toolbox is designed to allow for easy **customization**. Significant effort was made to modularize the code itself. It is also well-commented. It is therefore straightforward for a user to understand or modify the code to fit their needs. This is also the advantage of using a universal language such as `Python`.

Interpretation of Results - The Case Study In the case study, the modularity of the pipeline was particularly helpful. Only the preprocessing was run first. Results and ICA component images were checked, and artifacts were removed based on those checks. This first step was done for every subject. Then the feature extraction step was run. It was straightforward to resume from that point since previous results and objects—e.g., the filtered signal—were saved and could be

loaded directly. There was also uncertainty about the number of permutations to use for the SPoC significance assessment. This step—and only this step—was run multiple times with different numbers of permutations. It was very fast. Finally, the three SPoC component selection approaches could be tested independently without re-running earlier steps. However, the case study had some limitations. After the SPoC analysis, some noise remained in the patterns. The analysis also did not manage to automatically select the good SPoC components for all subjects, which complicated interpretation of the distance regression. The wireless EEG system introduced noise and interference compared to a wired cap. This may have degraded data quality.

Comparison with Existing Literature Compared with previously described toolboxes and pipelines—such as `NeuroKit2` [33], `NeuroPycon` [9] and `PREP` [4]—our toolbox is written in `Python` and provides a complete semi-automated and modular pipeline tailored for EEG decoding from raw data. As a wrapper of `MNE-Python`, it speeds up EEG analysis for researchers using this library.

Implications The developed toolbox has significant implications for the K3VR project and beyond. By providing a semi-automated pipeline for EEG decoding, as well as a real-time framework, this toolbox facilitates the data processing steps required to implement the final K3VR training and accelerates development. Beyond K3VR, the toolbox's modularity and adaptability make it suitable for a wide range of applications in brain–computer interfaces. The backbone of the pipeline—particularly the preprocessing steps that are common to many problems—can be reused for other tasks.

Future Work Excluding some subjects entirely when they lack relevant patterns could improve results, since outlier subjects influenced the SPoC alignment algorithm (approach 3). Implementing a hybrid of approach 3 (Hungarian algorithm, no baseline subject) and approach 2 (match best patterns first, noisy ones later) could be interesting to try and might yield better alignment results. Alternatively, manual component selection could be effective: a majority of subjects exhibited at least one relevant pattern in the parieto-occipital region, and manual selection would likely not take much time. This would ensure the distance regression is based on meaningful components; a fully automated SPoC component selection may have been too ambitious for this case study. Concerning the real-time framework, a wrapper could be implemented to improve usability by hiding low-level details. An experiment in real-time could then be imagined that way: (1) one calibration run to remove bad channels, remove artifacts, select SPoC components manually and save SSD and SPoC filters; (2) a second real-time run using the saved spatial filters to provide feedback based on the distance decoding. Another useful future work would be to create comprehensive documentation for the entire toolbox. Finally, the toolbox could be extended to handle additional data modalities, such as skin conductance response data.

6. Conclusion

This thesis addresses the challenge of developing a semi-automated, modular, and real-time-ready machine learning pipeline for EEG decoding, with a specific focus on assessing proximity in a virtual environment.

Methodology Summary The methodology involved several key steps. **Preprocessing** included resampling, artifact removal, re-referencing, filtering, and bad-channel interpolation to clean and standardize the EEG data. **Feature extraction** was performed using Spatio-Spectral Decomposition and Source Power Comodulation. **Component selection** explored three approaches to identify the most relevant SPoC components. Finally, a **real-time framework** was developed that simulated EEG processing via LSL to set the pipeline ready for real-time applications.

Contributions and Findings This thesis delivers three main contributions. First, a `Python`-based **toolbox** was developed that extends `MNE-Python` functions and machine learning algorithms into a semi-automated and modular pipeline. It allows users to run processing steps independently. Second, a novel approach for **SPoC component selection** based on the Hungarian algorithm was implemented. Third, a **real-time framework** was created to simulate EEG processing via LSL. Validation on EEG data from a virtual reality experiment showed that the preprocessing chain (resampling, artifact handling, re-referencing, filtering, interpolation) improved signal quality. SSD increased the signal-to-noise ratio for alpha-band activity, and SPoC identified components that comodulated with our target variable, the avatar distance. This cleaned signal allowed for distance assessment as well as proximity detection with an accuracy of up to 68%. The real-time framework sets this pipeline ready for real-time experiments.

Limitations and Generalizability Despite these strengths, the toolbox has practical limitations. It is focused on a single modality—EEG. Extending support to multimodal recordings would broaden applicability. In addition, adding a user-friendly wrapper to the real-time framework would simplify integration into user's code. Finally, even if the code is well commented and modularized, providing formal documentation and tutorials would improve accessibility and reproducible adoption.

Ethics The research conducted in this Master's thesis involved the collection and analysis of electroencephalography recordings. EEG data is considered sensitive personal data, as it provides insights into an individual's brain activity, cognitive

states, and potentially even emotional responses. As such, the ethical handling of this data is of paramount importance. All data collected during this study were anonymized. Each participant was assigned a unique identifier, and no personally identifiable information was stored alongside the EEG data. This data was not accessible to people outside the lab. Participants were informed about the data handling procedures, and their consent was obtained prior to data collection. All of this was approved by the local ethics committee. It is important to emphasize that the EEG data collected in this study were used exclusively for decoding distance and personal space intrusion in a controlled experimental setting, and not for interpreting or decoding broader cognitive or emotional states unrelated to the research objectives. Data was indeed not accessible for anything other than the specific analyses described in this thesis.

Lessons Learned Several lessons emerged from this work. First, while automation was desirable, certain steps such as bad channel removal and artifact removal have benefited from manual approaches. Second, coding in a modular way appeared to be crucial for the reusability of the code—even for me, for navigating and debugging. I improved my coding practices a lot. Finally, writing a thesis follows a specific format that I was not used to. I learned a lot while writing my own Master's thesis.

In conclusion, this work presents a significant step toward automating EEG data analysis. The developed toolbox and real-time framework provide a foundation for future research and applications in EEG-based neurophysiological analysis.

Bibliography

[1] Anijärv, Toomas Erik and Campbell, Alicia J. EEG-pyline: EEG pipeline in Python. *Zenodo* (2024)

[2] Appriou, A. and Pillette, L. and Trocellier, D. and Dutartre, D. and Cichocki, A. and Lotte, F. BioPyC, an Open-Source Python Toolbox for Offline Electroencephalographic and Physiological Signals Classification. *Sensors* (2021)

[3] Benwell, Christopher S.Y. and London, Robyn E. and Tagliabue, Chiara F. and Veniero, Domenica and Gross, Joachim and Keitel, Christian and Thut, Gregor. Frequency and power of human alpha oscillations drift systematically with time-on-task. *NeuroImage* (2019)

[4] Bigdely-Shamlo, Nima and Mullen, Tim and Kothe, Christian and Su, Kyung-Min and Robbins, Kay A. The PREP pipeline: standardized preprocessing for large-scale EEG analysis. *Frontiers in Neuroinformatics* (2015)

[5] Blankertz, Benjamin and Curio, Gabriel and Müller, Klaus-Robert. Classifying single trial EEG: towards brain computer interfacing. In: Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic. *MIT Press* (2001)

[6] Blankertz, Benjamin and Dornhege, Guido and Krauledat, Matthias and Müller, Klaus-Robert and Curio, Gabriel. The non-invasive Berlin Brain–Computer Interface: Fast acquisition of effective performance in untrained subjects. *NeuroImage* (2007)

[7] Blankertz, Benjamin and Losch, Florian and Krauledat, Matthias and Dornhege, Guido and Curio, Gabriel and Müller, Klaus-Robert. The Berlin Brain-Computer Interface: Accurate performance from first-session in BCI-naive subjects. *IEEE Transactions on Biomedical Engineering* (2008)

[8] Breunig, Markus M. and Kriegel, Hans-Peter and Ng, Raymond T. and Sander, Jörg. LOF: identifying density-based local outliers. *Association for Computing Machinery* (2000)

[9] David Meunier and Annalisa Pascarella and Dmitrii Altukhov and Mainak Jas and Etienne Combrisson and Tarek Lajnef and Daphné Bertrand-Dubois and Vanessa Hadid and Golnoush Alamian and Jordan Alves and Fanny Barlaam and Anne-Lise Saive and Arthur Dehgan and Karim Jerbi.

NeuroPycon: An open-source python toolbox for fast multi-modal and reproducible brain connectivity pipelines. *NeuroImage* (2020)

[10] Davis, Bruce M. Uses and abuses of cross-validation in geostatistics. *Mathematical Geology* (1987)

[11] Debnath, Ranjan and Buzzell, George A. and Morales, Santiago and Bowers, Maureen E. and Leach, Stephanie C. and Fox, Nathan A. The Maryland analysis of developmental EEG (MADE) pipeline. *Psychophysiology* (2020)

[12] Donoghue, Thomas and Haller, Matar and Peterson, Erik J. and Varma, Paroma and Sebastian, Priyadarshini and Gao, Richard and Noto, Torben and Lara, Antonio H. and Wallis, Joni D. and Knight, Robert T. and Shestyuk, Avgusta and Voytek, Bradley. Parameterizing neural power spectra into periodic and aperiodic components. *Nature Neuroscience* (2020)

[13] Dähne, Sven and Meinecke, Frank and Haufe, Stefan and Höhne, Johannes and Tangermann, Michael and Müller, Klaus-Robert and Nikulin, Vadim V. SPoC: A novel framework for relating the amplitude of neuronal oscillations to behaviorally relevant parameters. *NeuroImage* (2014)

[14] Fazli, Siamac and Grozea, Cristian and Danoczy, Marton and Blankertz, Benjamin and Popescu, Florin and Müller, Klaus-Robert. Subject independent EEG-based BCI decoding. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., Culotta, A. (eds.) Advances in Neural Information Processing Systems. *Curran Associates, Inc* (2009)

[15] Gil Ávila, Cristina and Bott, Felix S. and Tiemann, Laura and Hohn, Vanessa D. and May, Elisabeth S. and Nickel, Moritz M. and Zebhauser, Paul Theo and Gross, Joachim and Ploner, Markus. DISCOVER-EEG: an open, fully automated eeg pipeline for biomarker discovery in clinical neuroscience. *Scientific Data* (2023)

[16] Graeler, Benedikt and Gerharz, Lydia and Pebesma, Edzer. Spatio-temporal analysis and interpolation of PM 10 measurements in Europe. *European Topic Centre on Air Pollution and Climate Change Mitigation* (2013)

[17] Gramfort, Alexandre and Luessi, Martin and Larson, Eric and Engemann, Denis A. and Strohmeier, Daniel and Brodbeck, Christian and Goj, Roman and Jas, Mainak and Brooks, Teon and Parkkonen, Lauri and Hämäläinen, Matti. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience* (2013)

[18] Harmeling, Stefan and Dornhege, Guido and Tax, David and Meinecke, Frank and Müller, Klaus-Robert. From outliers to prototypes: Ordering data. *Neurocomputing* (2006)

[19] Haufe, Stefan and Meinecke, Frank and Görgen, Kai and Dähne, Sven and Haynes, John-Dylan and Blankertz, Benjamin and Bießmann, Felix. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage* (2014)

[20] Hayduk, Leslie A. Personal space: An evaluative and orienting overview. *Psychological Bulletin* (1978)

[21] Hofmann, Simon M. and Klotzsche, Felix and Mariola, Alberto and Nikulin, Vadim and Villringer, Arno and Gaebler, Michael. Decoding subjective emotional arousal from EEG during an immersive virtual reality experience. *eLife* (2021)

[22] Iachini, Tina and Coello, Yann and Frassinetti, Francesca and Senese, Vincenzo Paolo and Galante, Francesco and Ruggiero, Gennaro. Peripersonal and interpersonal space in virtual and real environments: Effects of gender and age. *Journal of Environmental Psychology* (2016)

[23] Jiang, X. and Bian, G. B. and Tian, Z. Removal of Artifacts from EEG Signals: A Review. *Sensors* (2019)

[24] K3VR Project Consortium. K3VR: De-escalating Conflicts and Crises through Communication. `https://www.k3vr.de/`, accessed: 2025-10-12 (2023)

[25] Kothe, C., Shirazi, S.Y., Stenner, T., Medine, D., Boulay, C., Grivich, M.I., Artoni, F., Mullen, T., Delorme, A., Makeig, S.. The Lab Streaming Layer for Synchronized Multimodal Recording. *Imaging Neuroscience* (2025)

[26] Krell, Mario M. and Straube, Sirko and Seeland, Anett and Wöhrle, Hendrik and Teiwes, Johannes and Metzen, Jan H. and Kirchner, Elsa A. and Kirchner, Frank. pySPACE—a signal processing and classification environment in Python. *Frontiers in Neuroinformatics* (2013)

[27] Kuhn, Harold W. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly* (1955)

[28] Kumaravel, Velu Prabhakar and Buiatti, Marco and Parise, Eugenio and Farella, Elisabetta. Adaptable and Robust EEG Bad Channel Detection Using Local Outlier Factor (LOF). *Sensors* (2022)

[29] Lepot, Mathieu and Aubin, Jean-Baptiste and Clemens, François H.L.R. Interpolation in Time Series: An Introductive Overview of Existing Methods, Their Performance Criteria and Uncertainty Assessment. *Water* (2017)

[30] Litkouhi, S. and Geramipour, M. and Litkouhi, S. The Effect of Gender, Age, and Nationality on the Personal Space Preferences in Children's Hospitals among Iranian and German Children and Adolescents. *Iranian Red Crescent Medical Journal* (2012)

[31] Liu, Q. and Balsters, J. H. and Baechinger, M. and van der Groen, O. and Wenderoth, N. and Mantini, D. Estimating a neutral reference for electroencephalographic recordings: the importance of using a high-density montage and a realistic head model. *Journal of Neural Engineering* (2015)

[32] Luck, Steven. An Introduction to The Event-Related Potential Technique. *MIT Press* (2005)

[33] Makowski, Dominique and Pham, Tam and Lau, Zitong and Chen, S.H. Annabel and Lombardi, Fabio and Blum, Christian and Schroeder, Eric and Tissot, Paul and Aghababaei, Samavati and Mehta, Neeraj and others. NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behavior Research Methods* (2021)

[34] MNE Developers. Algorithms and other implementation details. `https://mne.tools/stable/documentation/implementation.html#bad-channel-repair-via-interpolation`, accessed: 2025-09-25 (2025)

[35] MNE Developers. Artifact Overview. `https://mne.tools/stable/auto_tutorials/preprocessing/10_preprocessing_overview.html#tut-artifact-overview`, accessed: 2025-09-19 (2025)

[36] MNE Developers. Independent Component Analysis (ICA). `https://www.nmr.mgh.harvard.edu/mne/0.14/manual/preprocessing/ica.html`, accessed: 2025-09-19 (2025)

[37] Motamedi-Fakhr, Shayan and Moshrefi-Torbati, M. and Hill, Martyn and Hill, Catherine and White, Paul. Signal processing techniques applied to human sleep EEG signals - a review. *Biomedical Signal Processing and Control* (2014)

[38] Müller, Klaus-Robert and Tangermann, Michael and Dornhege, Guido and Krauledat, Matthias and Curio, Gabriel and Blankertz, Benjamin. Machine learning for real-time single-trial EEG-analysis: From brain–computer interfacing to mental state monitoring. *Journal of Neuroscience Methods* (2008)

[39] Nikulin, Vadim V. and Nolte, Guido and Curio, Gabriel. A novel method for reliable and fast extraction of neuronal EEG/MEG oscillations on the basis of spatio-spectral decomposition. *NeuroImage* (2011)

[40] Pedregosa, Fabian and Varoquaux, Gaël and Gramfort, Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel, Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss, Ron and Dubourg, Vincent and Vanderplas, Jake and Passos, Alexandre and Cournapeau, David and Brucher, Matthieu and Perrot, Matthieu and Duchesnay, Édouard. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* (2011)

[41] Perrin, F. and Pernier, J. and Bertrand, O. and Echallier, J.F. Spherical splines for scalp potential and current density mapping. *Electroencephalography and Clinical Neurophysiology* (1989)

[42] Peter D. Welch. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics* (1967)

[43] Pieper, Kerstin and Spang, Robert P. and Prietz, Pablo and Möller, Sebastian and Paajanen, Erkki and Vaalgamaa, Markus and Voigt-Antons, Jan-Niklas . Working With Environmental Noise and Noise-Cancelation: A Workload Assessment With EEG and Subjective Measures. *Frontiers in Neuroscience* (2021)

[44] Pion-Tonachini, Laura and Kreutz-Delgado, Kenneth and Makeig, Scott. ICLabel: An automated electroencephalographic independent component classifier, dataset, and website. *Neuroimage* (2019)

[45] Richardson, Michael and Jicol, Cristina and Taulo, Gabriel and Park, Juyoung and Kim, Hyun Kyung and Proulx, Michael J. and de Sousa, Alex A. Differences in office-based personal space perception between British and Korean populations. *Frontiers in Psychology* (2023)

[46] Risk, C and James, P. M. A. Optimal cross-validation strategies for selection of spatial interpolation models for the Canadian Forest Fire Weather Index system. *Earth and Space Science* (2022)

[47] Ruggiero, Giovanni and Frassinetti, Francesca and Coello, Yann and et al. The effect of facial expressions on peripersonal and interpersonal spaces. *Psychological Research* (2017)

[48] Ryali, S. and Glover, G. H. and Chang, C. and Menon, V. Development, validation, and comparison of ICA-based gradient artifact reduction algorithms for simultaneous EEG-spiral in/out and echo-planar fMRI recordings. *NeuroImage* (2009)

[49] Satopaa, Ville and Albrecht, Jeannie and Irwin, David and Raghavan, Barath. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. *2011 31st International Conference on Distributed Computing Systems Workshops* (2011)

[50] Scheltienne, M., Larson, E., Desvachez, A., Lee, K.. MNE-LSL: Real-time framework integrated with MNE-Python for online neuroscience research through LSL-compatible devices. *Journal of Open Source Software* (2025)

[51] Schirrmeister, Robin Tibor and Springenberg, Jost Tobias and Fiederer, Lukas Dominique Josef and Glasstetter, Martin and Eggensperger, Katharina and Tangermann, Michael and Hutter, Frank and Burgard,

Wolfram and Ball, Tonio. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping* (2017)

[52] Scikit-learn Developers. scikit-learn: Leave-One-Out Cross-Validation. `https://scikit-learn.org/stable/modules/cross_validation.html#leave-one-out-loo`, accessed: 2025-09-02 (2025)

[53] Smith, EE and Reznik, SJ and Stewart, JL and Allen, JJ. Assessing and conceptualizing frontal EEG asymmetry: An updated primer on recording, processing, analyzing, and interpreting frontal alpha asymmetry. *International Journal of Psychophysiology* (2017)

[54] Tomczak, Maciej. Spatial Interpolation and its Uncertainty Using Automated Anisotropic Inverse Distance Weighting (IDW) - Cross-Validation/Jackknife Approach. *Journal of Geographic Information and Decision Analysis* (1998)

[55] Tootell, Roger B. H. and Zapetis, Sarah L. and Babadi, Baktash and et al. Psychological and physiological evidence for an initial 'Rough Sketch' calculation of personal space. *Scientific Reports* (2021)

[56] Venthur, B., Dähne, S., Höhne, J., Heller, H., Blankertz, B.. Wyrm: A Brain-Computer Interface Toolbox in Python. *Neuroinformatics* (2015)

[57] Venthur, Bastian and Blankertz, Benjamin. Mushu, a free- and open source BCI signal acquisition, written in Python. *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2012)

[58] Venthur, Bastian and Scholler, Stefan and Williamson, John and Dahne, Stephan and Treder, Matthias S. and Kramarek, Michael T. and Müller, Klaus-Robert and Blankertz, Benjamin. Pyff—a pythonic framework for feedback applications and stimulus presentation in neuroscience. *Frontiers in Neuroinformatics* (2010)

[59] Vorwerk, J. and Oostenveld, R. and Piastra, M.C. and Kocsis, L. and Wolters, C.H. The FieldTrip-SimBio pipeline for EEG forward solutions. *BioMedical Engineering OnLine* (2018)

[60] Winkler, Irene and Debener, Stefan and Müller, Klaus-Robert and Tangermann, Michael. On the influence of high-pass filtering on ICA-based artifact reduction in EEG-ERP. *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2015)

[61] Zhang, Guanghui and Garrett, David and Luck, Steven. Optimal filters for ERP research II: Recommended settings for seven common ERP components. *Psychophysiology* (2024)

A. Appendix. Spherical Spline Interpolation

Here is a description of the spherical spline interpolation, as described in the `MNE-Python` documentation *Algorithms and other implementation details* [34].

Spherical-spline model $V(\mathbf{r}_i)$ is the potential at position $\mathbf{r}_i$ on the unit sphere:

$$V(\mathbf{r}_i) = c_0 + \sum_{j=1}^{N} c_j \, g_m\big(\cos(\mathbf{r}_i, \mathbf{r}_j)\big), \tag{A.1}$$

where $\mathbf{C} = (c_1, \ldots, c_N)^\top$ are constants to be estimated, and $g_m(\cdot)$ is the spline kernel of order m:

$$g_m(x) = \frac{1}{4\pi} \sum_{n=1}^{\infty} \frac{2n+1}{\big(n(n+1)\big)^m} \, P_n(x), \tag{A.2}$$

with $P_n(x)$ the Legendre polynomials.

Estimating the constants The constants $\mathbf{C}$ are obtained by solving:

$$\begin{bmatrix} c_0 \\ \mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_s^\top & 0 \\ \mathbf{T}_s & \mathbf{G}_{ss} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{X} \end{bmatrix} \quad = \quad \mathbf{C}_1 \, \mathbf{X}. \tag{A.3}$$

where: $\mathbf{X} \in \mathbb{R}^{N \times 1}$ is the vector of good-channel data, $\mathbf{G}_{ss} \in \mathbb{R}^{N \times N}$ is the kernel matrix with $[\mathbf{G}_{ss}]_{ij} = g_m\big(\cos(\mathbf{r}_i, \mathbf{r}_j)\big)$, $\mathbf{T}_s = (1, 1, \ldots, 1)^\top \in \mathbb{R}^{N \times 1}$, and $\mathbf{C}_1$ is $\begin{bmatrix} \mathbf{T}_s^\top & 0 \\ \mathbf{T}_s & \mathbf{G}_{ss} \end{bmatrix}^{-1}$ without its first column.

Interpolating bad channels The interpolated potentials $\hat{\mathbf{X}}$ are then obtained with:

$$\hat{\mathbf{X}} = \mathbf{G}_{ds} \, \mathbf{C} + \mathbf{T}_d \, c_0, \tag{A.4}$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{M \times 1}$ are the interpolated potentials at the M bad channels, $\mathbf{G}_{ds} \in \mathbb{R}^{M \times N}$ is the kernel matrix with $[\mathbf{G}_{ds}]_{ij} = g_m\big(\mathbf{r}_i, \mathbf{r}_j\big)$, comparing each bad channel position $\mathbf{r}_i$ to each good channel position $\mathbf{r}_j$, $\mathbf{C}$ are the constants estimated from the good channels, $\mathbf{T}_d = (1, 1, \ldots, 1)^\top \in \mathbb{R}^{M \times 1}$, and c_0 is the constant offset term.

From (A.3) and (A.4), we have:

$$\hat{\mathbf{X}} \;=\; \big[\, \mathbf{T}_d \;\; \mathbf{G}_{ds} \,\big] \begin{bmatrix} c_0 \\ \mathbf{C} \end{bmatrix} \;=\; \underbrace{\big[\, \mathbf{T}_d \;\; \mathbf{G}_{ds} \,\big] \, \mathbf{C}_1}_{\text{mapping matrix}} \, \mathbf{X} \tag{A.5}$$

B. Appendix. SPoC activation patterns

Figure 39: SPoC activation patterns for (a) subject 45, (b) subject 123, (c) subject 179, (d) subject 474, (e) subject 231, (f) subject 451, (g) subject 570, (h) subject 973.

C. Appendix. SPoC patterns alignment, with approach 2

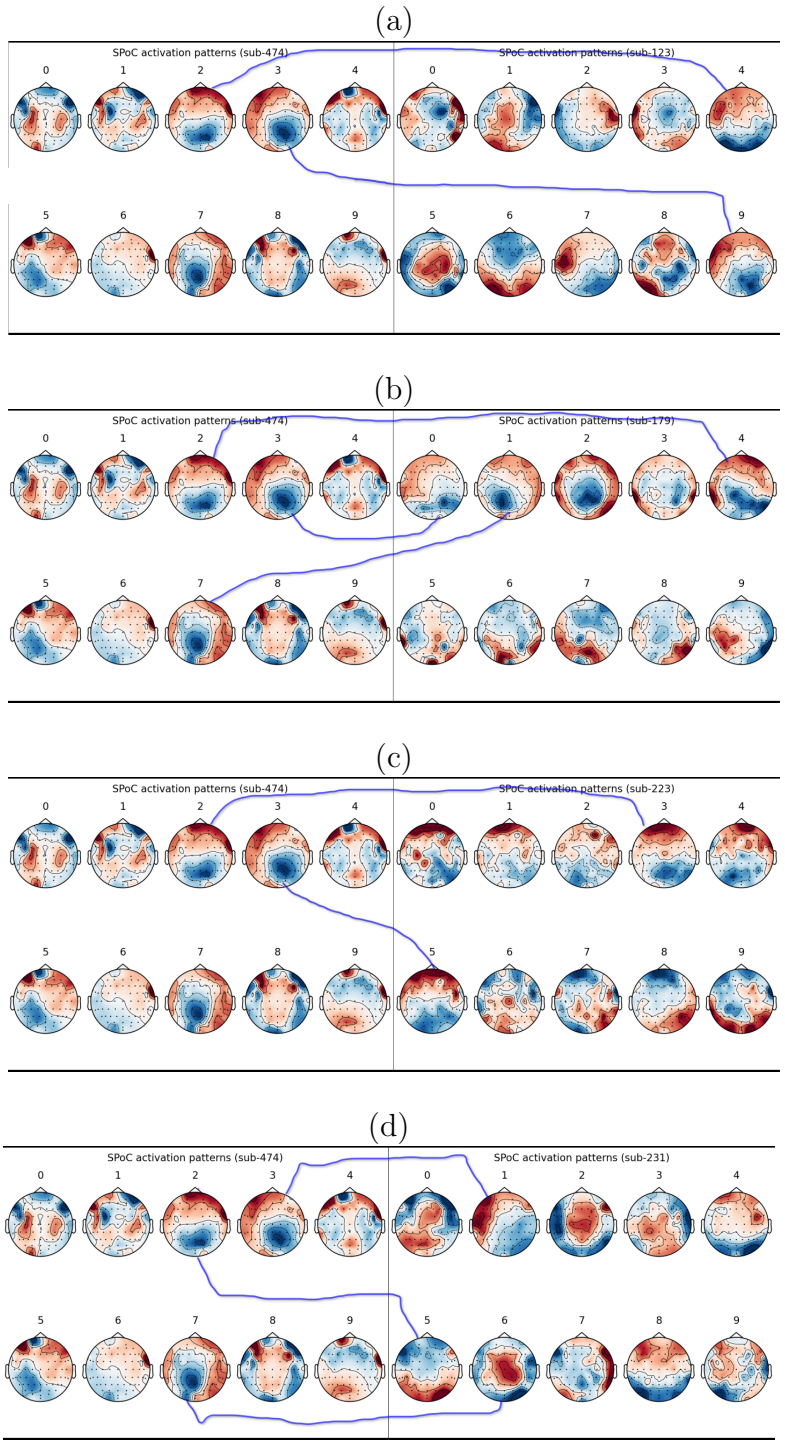


Figure 40: Aligned SPoC activation patterns—using approach 2 with subject 474 as baseline subject— for (a) subject 123, (b) subject 179, (c) subject 223, and (d) subject 231. The best 2 or 3 matching patterns are shown, with a blue link.

D. Appendix. SPoC patterns alignment, with approach 3

Global Pattern	sub-045	sub-096	sub-121	sub-123	sub-179	sub-223	sub-231	sub-298
0	5	5	1	1	1	5	0	5
1	9	1	5	6	5	1	1	6
2	1	6	8	5	2	2	8	2
3	0	9	7	7	0	0	3	0
4	3	8	2	3	9	8	5	8
5	6	0	0	2	3	6	6	4
6	8	7	6	0	8	7	7	7
7	7	2	9	9	7	9	9	9
8	4	4	4	4	4	4	4	1
9	2	3	3	8	6	3	2	3

Table 4: Indices of patterns to be aligned with the global ones, for each subject (1).

Global Pattern	sub-300	sub-390	sub-444	sub-451	sub-474	sub-482	sub-570	sub-973
0	7	1	5	7	2	2	4	1
1	9	9	1	1	7	7	9	2
2	2	3	8	8	6	3	6	8
3	5	7	2	5	1	9	0	5
4	8	8	0	0	5	8	5	0
5	1	6	6	9	4	1	1	6
6	0	0	7	6	0	0	2	7
7	6	2	9	2	9	6	7	9
8	4	4	4	4	8	4	8	4
9	3	5	3	3	3	5	3	3

Table 5: Indices of patterns to be aligned with the global ones, for each subject (2).