UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

**Lari Kuistio**
**Joona Meriläinen**

# USE OF INDEPENDENT COMPONENT ANALYSIS FOR HEAD MOVEMENT ARTIFACT DETECTION IN EEG

Bachelor's Thesis
Degree Programme in Computer Science and Engineering
February 2023

**Kuistio L., Meriläinen J. (2023) Use of Independent Component Analysis for Head Movement Artifact Detection in EEG.** University of Oulu, Degree Programme in Computer Science and Engineering, 54 p.

## ABSTRACT

**EEG data is often contaminated with artifacts and noise from various sources, such as eye blinks, jaw movement and AC outlets. VR opens up many new interesting opportunities for EEG research. However, VR usually involves head movement, which can also cause notable artifacts in EEG data. We need an effective method to remove the head movement related artifacts, so that VR can be effectively used in EEG experiments. In this study, we apply independent component analysis (ICA) for removing head movement artifacts from EEG data. We constructed an experiment where a subject is placed into a VR environment that incentivizes them to move their head. At the same time, the subject is performing a conventional auditory oddball experiment, which is known to cause an ERP containing a measurable P300 component. We attempt to remove head movement related artifacts from the data without removing ERP components of interest, such as the P300, as a side effect. Our data processing pipeline was implemented using Matlab with the EEGLAB and ERPLAB toolboxes, along with AMICA as our chosen ICA implementation. We explain the design and implementation of both the experiment and the following data processing. We then discuss the results and how they could help future research. We were able to distinguish head movement related components with ICA, but the impact of their removal ended up being fairly limited. We also found out that the head movements seem to have an effect on the shape and amplitude of the P300 component.**

**Keywords: EEG, ERP, independent component analysis, VR, oddball, head movement artifacts**

**Kuistio L., Meriläinen J. (2023) Independent Component Analysis -menetelmän käyttö pään liikkeistä aiheutuvien häiriöiden havaitsemiseen EEG:ssä.** Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 54 s.

## TIIVISTELMÄ

**EEG-data sisältää tyyppillisesti useista eri lähteistä peräisin olevia häiriöitä. Näitä lähteitä voivat olla esimerkiksi silmänräpäytykset, leuan liikkeet sekä koetilassa olevat vaihtovirtaa kuljettavat sähköjohdot ja pistorasiat. VR avaa uusia mielenkiintoisia mahdollisuuksia EEG-tutkimuksia varten, mutta VR-ympäristöissä koehenkilö tyypillisesti liikuttelee päätään, mikä voi aiheuttaa merkittäviä häiriöitä EEG-dataan. Tämän takia tarvitaan menetelmä poistamaan pään liikkeistä aiheutuvia häiriöitä, jotta VR:n käyttö voi yleistyä EEG-tutkimuksissa. Tässä tutkimuksessa pyrimme käyttämään ICA-menetelmää pään liikkeistä aiheutuvien häiriöiden poistamiseksi. Rakensimme koeasetelman, jossa koehenkilö asetetaan VR-ympäristöön, joka kannustaa häntä liikuttamaan päätään. Samaan aikaan koehenkilö suorittaa tavanomaista "oddball-koetta", jonka tiedetään aiheuttavan P300-komponentin sisältävän ERP:n. Pyrimme poistamaan pään liikkeistä aiheutuvat häiriöt ilman että poistamme samalla kertaa mahdollisesti kiinnostavia komponentteja, kuten P300:n. Data-analyysin toteuttamiseen käytimme Matlab-ohjelmistoa yhdessä EEGLAB- ja ERPLAB-lisäosien kanssa. Käyttämämme ICA-algoritmi oli AMICA. Kerromme sekä koeasetelman että sitä seuranneen data-analyysivaiheen suunnittelusta ja toteutuksesta. Lisäksi käymme läpi tutkimuksen tulokset ja pohdimme miten ne voivat auttaa tutkimustyötä tulevaisuudessa. Onnistuimme erottelemaan pään liikkeisiin liittyviä komponentteja ICA-menetelmän avulla, mutta niiden poistamisella oli hieman odotettua pienempi vaikutus. Havaitsimme myös, että pään liikkeillä näyttää olevan jonkinlaista vaikutusta P300-komponentin muotoon ja amplitudiin.**

**Avainsanat: aivosähkökäyrä, eeg, erp, ica, VR**

# TABLE OF CONTENTS

# FOREWORD

This paper is our bachelor's thesis for the Degree Programme in Computer Science and Engineering in the University of Oulu. We would like to thank our supervisor Dr. Evan Center for assisting with this project. We would also like to thank Timo Ojala who hosted the Applied Computing Project course, in which the topic of this paper was assigned to us.

Oulu, February 8th, 2023

Lari Kuistio
Joona Meriläinen

# LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| CCA | Canonical-correlation analysis |
| EEG | Electroencephalography |
| ERP | Event-related potential |
| HMD | Head-mounted display |
| ICA | Independent component analysis |
| IVA | Independent vector analysis |
| VE | Virtual environment |
| VR | Virtual reality |

# 1. INTRODUCTION

In 1929, a German physiologist and psychiatrist Hans Berger published a set of experiments in which he demonstrated that the electrical activity of the human brain could be measured and recorded by placing an electrode on the scalp, amplifying the signal, and plotting the changes in voltage over time. These findings lead to a method that we nowadays know as electroencephalography, or EEG [1].

During the following decades, EEG was found to be useful in many scientific applications [2]. Researchers discovered that embedded within the EEG, there are neural responses associated with specific sensory, cognitive, and motor events. These specific responses, event-related potentials, could then be extracted from the data. Today, researchers understand that these ERPs provide high-resolution temporal information about the mind and brain that cannot be obtained any other way, which has lead to the growth of the field of ERP research [3].

However, in its raw unprocessed form, EEG is a fairly crude measurement of brain activity. EEG is a mixture of countless different neural sources, making it impossible to isolate specific neuro-cognitive processes. The unfortunate reality is that other electrical activity from the body and the environment can contaminate the measured signal. EEG data is prone to a multitude of so-called artifacts that are caused by natural human behaviour, such as eye movement, blinking and muscle activity. For example, a typical eye blink can be 100 times larger than many ERP experimental effects. The artifacts and noise can easily drown out the actual interesting EEG signal. The solution is to either prevent them during the data collection session or remove them from the data afterwards via software [1].

Over time, science has progressed and new methods for research have emerged. One notable example being the use of virtual reality [4]. The use of VR and EEG has been proven as a valid method of conducting experiments with plenty of potential for the future [5]. However, the use of VR in ERP research can often lead to the previously mentioned issue where the data gets contaminated by noise and artifacts.

In ERP research, many have opted to restrict the behavior of the subject in order to avoid causing too many artifacts in the data. But this is a very limiting option when using VR as a part of the experiment. In order to have as accurate and realistic results as possible, limitations need to be set to a minimum. This leads to finding ways to effectively post-process the data to clean up the unwanted artifacts and noise without affecting the underlying brain activity data.

One relatively new approach for artifact correction is the use of independent component analysis (ICA). It is a statistical and computational technique for revealing hidden factors that underlie sets of random variables, measurements, or signals. It is a very powerful technique which is capable of finding underlying sources when classic methods fail completely. ICA attempts to decompose a multivariate signal into independent signals in order to enable a separation of the unwanted signals [6].

Since EEG is a mixture of signals, ICA manages to identify and eliminate artifacts in the data, without altering the actual brain activity. This method has been proven to be successful before. Particularly, high voltage artifacts such as eye movements have been proven to be easily removed [7].

In the experimental study covered in this paper, we aim to discover if ICA can also isolate components related head motions caused by interacting with a VR environment.

The ability to effectively isolate these components would help with a design of a comprehensive post-processing pipeline for EEG data. This would then promote the idea of having less restricted experiments with VR and EEG in the future.

# 2. RELATED WORK

## 2.1. EEG

A book called "An Introduction to the Event-Related Potential Technique" by Steven J. Luck [1] and its 2nd edition [3] are a great introduction to the world of EEG and ERP technique. In addition, "Foundations of Signal Processing" by Vetterli, Kovacevic and Goyal [8] is a good resource for basic signal processing methods and concepts. These two books, and especially Luck's books, provide a basic body of knowledge and a foundation for our study.

## 2.2. VR and EEG

VR and EEG have been successfully combined in multiple previous studies. The studies have been conducted on various topics, such as driving behavior (Bayliss & Ballard, 2000), spatial navigation (e.g., Bischof & Boulanger, 2003), and spatial presence (Baumgartner, Valko, Esslen, & Jänke, 2006)[5].

Krokos and Varshney [9] studied the application of EEG for quantifying VR cybersickness and found out that certain EEG activities are associated with VR cybersickness. In addition, Grassini et. al. [10] used EEG and ERP to evaluate the sense of presence in an immersive virtual environment, and found out that certain ERP components correlate with a sense of presence reported through questionnaires.

According to Bohil et. al. [4], VR is being increasingly used by neuroscientists because it provides ways to simulate natural events and social interactions while maintaining a relatively good level of control over the research environment. Keeping the environment and all possible variables under control, while at the same time trying to make the experience as natural as possible for the test subject, can be difficult in traditional experiments [4].

Baumgartner et. al. [11] successfully used noninteractive VR to study the neurophysiological underpinnings of spatial presence as early as 2006. Also, Bischof et. al. [12] used VR and EEG to study spacial navigation.

All these examples show that VR and EEG has been successfully combined in many previous studies. But there are still many topics to experiment with, so the possibilities for the future are very broad.

## 2.3. Artifacts in EEG

"A new method for off-line removal of ocular artifact" [13] is a classic paper related to so called ocular artifacts: artifacts caused by eye movements and blinks. The paper mentions that during its time, a common procedure to remove ocular artifacts was to simply discard all the data that has those artifacts in them. However, this is not always possible, because eye movements and blinks may have an important connection with the underlying cognitive processes that are under study. Some subjects may also be unable to consciously avoid eye movements for various reasons. Furthermore, telling test subjects to keep their eyes still and not blink effectively assigns them an additional task, which may take focus away from something else. The paper describes a method that essentially tries to remove the disturbances caused by eye movements by subtracting the disturbances from the EEG data.

A paper by Ochoa and Polich supports the idea that asking subjects not to blink assign them a secondary task that changes the resulting EEG signal [14]. They found out that asking subjects not to blink results in a decreased P300 amplitude. The fact that this kind of an effect can be observed in the ERP suggests that asking subjects not to blink does in some way affect the underlying cognitive processes. This is generally something that we do not want to happen, because we want the EEG signal to reflect the phenomena under study instead of these sort of superfluous aspects.

Regan et. al. [15] support the notion that detecting and removing artifacts in EEG signals is important. They point out that this is important in ambulatory EEG applications where limiting movement can be almost impossible. They suggest a method based on Mutual Information Evaluation Function and Linear Discriminant Analysis to detect head movement artifacts. However, instead of trying to remove the artifacts from the EEG data, they only focus on classifying them. Christoph et. al. [16] studied the suppression of head movement artifacts in a VR setting, which is very close to what we are trying to do in our study. They note that countering movement artifacts is crucial for EEG to be a viable measurement tool in VR experiments, because VR usually inherently involves some amount of movement. The method for removing artifacts they investigate is a bit unclear, but it is based on a warp correlation filter.

Mumtaz et. al. [2] reviewed the different challenges commonly faced when dealing with EEG artifact removal methods. In simple terms, the main challenge with artifact removal methods is that they usually remove something else in addition to the artifact itself. This can cause valuable data to be lost. Chen et. al. [17] reviewed some different methods for removing muscle artifacts from EEG signal. According to their paper, traditional methods include filtering and linear regression. In addition to these traditional methods, there are source separation algorithms and single channel signal decomposition techniques. Source separation algorithms include ICA (independent component analysis), CCA (canonical correlation analysis) and IVA (independent vector analysis). Single channel signal decomposition techniques include prior-knowledge-based signal decomposition and data-driven signal decomposition methods. In our own study, we are going to look at ICA.

## 2.4. Independent Component Analysis

The 2002 article "Independent component analysis: an introduction" [18] gives an introduction to the ICA method. The article is not related to EEG specifically but gives a good introduction to what ICA is. ICA is one case of blind source separation, which attempts to decompose a signal formed through the linear combination of signals from different sources, into the original source signals [19]. Blind source separation tries to achieve this with no knowledge of the structure of the original source signals or their linear combination [19]. ICA is based on the assumption that the source signals are uncorrelated, statistically independent and non-Gaussian [18]. Makeig and Orton [20] give an overview of ICA's relation to specifically EEG and ERP. Hoffman and Falkenstein [21] compare ICA and EMCP. EMCP is the technique that was used in the

1983 paper "A new method for off-line removal of ocular artifact" [13]. EMCP stands for "eye movement correction procedure" and it removes eye-movements by regressing them away. Urigüen et. al. [22] found out that if there is no prior knowledge about the EEG signal and the types of artifacts it will include, then ICA is the safest option for removing artifacts from EEG data. According to Xue et. al. [23], ICA is an effective method for removing eye-blink artifacts and also power line noise. We therefore expect it to also work for other muscle sources that are statistically independent from brain sources.

# 3. DESIGN

In this chapter, we discuss the design of our data collection experiment and cover the main principles of how we approached the post-processing of the collected data.

## 3.1. Data Collection Experiment

Since the plan for this study was to experiment with removing unwanted artifacts in EEG data, we needed to collect suitable data that we could then apply ICA to. And since our focus is on head movement motivated by the use of VR, it was obvious that we needed to design an experiment that utilizes VR. The virtual reality environment then needed to lead the subject to move their head and eyes in a natural way. Our goal was to to create a scenario where the head and eye movements would not be the main focus of the subject, but instead come somewhat naturally. This meant that the VR environment should contain something that would cause the subject to move their head and eyes without having to specifically focus on it. This is a more realistic scenario than if we would, for example, just ask the subject to move their head with no associated stimulus.

### *3.1.1. Experiment Setup*

Since we wanted to record EEG while using VR, we needed to construct a suitable experiment setup with all the necessary equipment. Our university was generous enough to provide all the required equipment for us. Here we will briefly describe the experiment setup.

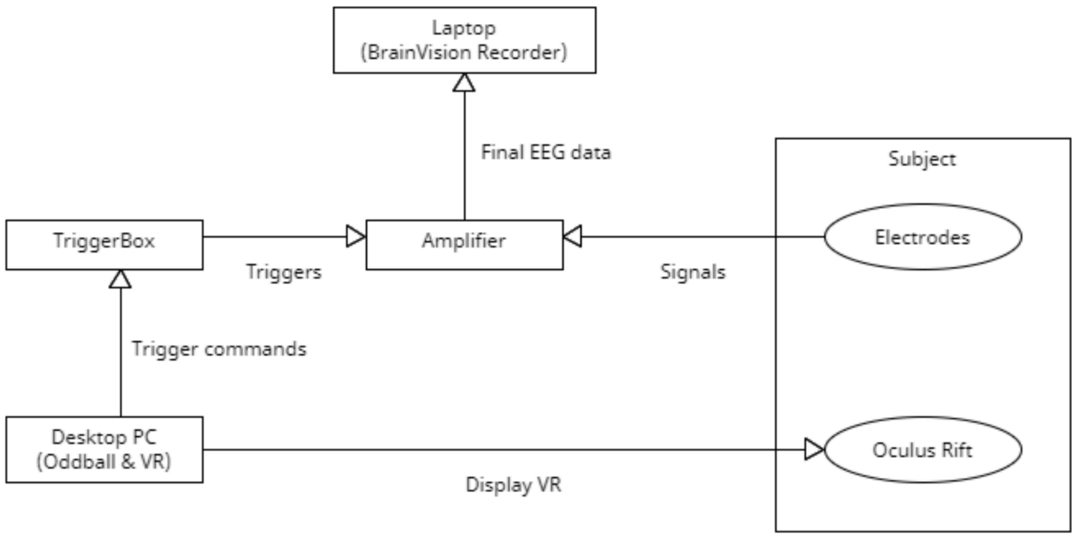

Figure 1. Diagram of the complete experiment setup.

To record the EEG signal from the subject’s scalp, suitable electrodes and a cap to hold them in place are needed. For the electrodes and cap, we used a Brain Products actiCAP snap and the included silver/silver chloride (Ag/AgCl) electrodes. The cap includes slots for the electrodes. 32 electrodes in total were used. Most of the electrode locations adhered to the international 10-20 system. Those electrodes included Fp1, Fz, F3, F7, FC5, FC1, C3, M1, CP5, CP1, Pz, P3, P7, O1, Oz, O2, P4, P8, M2, CP6, CP2, Cz, C4, FC6, FC2, F4, F8 and Fp2. In addition, we used four other electrodes. Two for above and below the left eye, and two for the left and right temples. We used the Cz electrode as the online reference. This was due to it being located near the center of the scalp and us being interested in the activity related to head motion that presumably happens near neck muscles. Figure 2 shows an illustration of most of the electrode locations on the scalp.

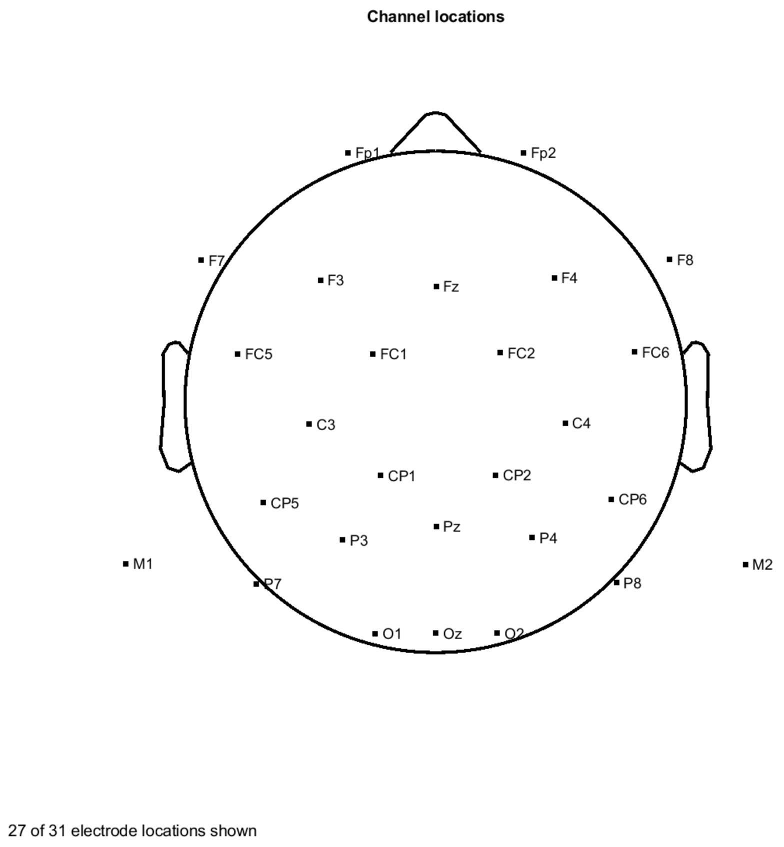

Figure 2. Electrode locations on the scalp.

The electrodes were connected to a Brain Products ActiCHamp Plus, which is a 24-bit amplifier. The amplifier was connected to a laptop running the BrainVision Recorder software, which is used to record the EEG data and also to measure the electrodes' impedances during the setup phase. The amplifier records the data at a 1000 Hz sampling frequency.

We also used an additional device called a TriggerBox to create markers in the data when the auditory stimulus is given or the subject turns their head. The device in use was the Brain Product's TriggerBox. It was connected to the amplifier and to the computer where the virtual reality environment and the software used to provide the auditory stimulus were running.

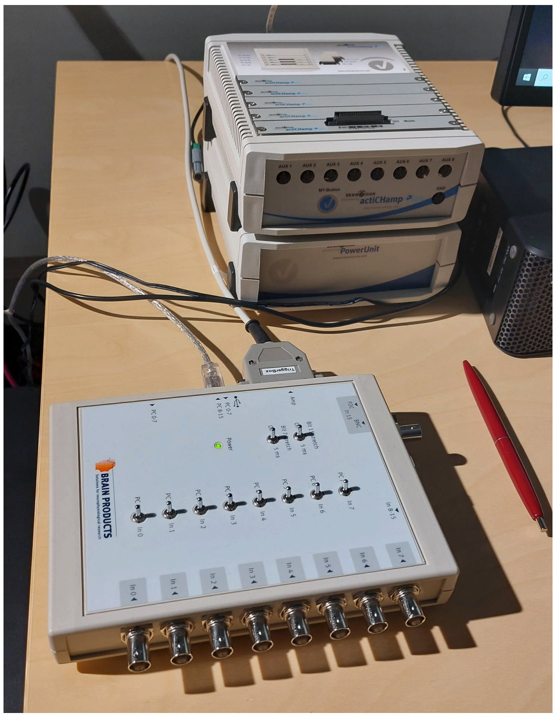

Figure 3. Amplifier (further back) and TriggerBox (in front).

### *3.1.2. Oddball Experiment*

In addition to the artifacts, the EEG data should contain brain activity that is unrelated to the artifacts. This is required for us to be able to investigate whether our noise and artifact removal was successful or not. Without any interesting data, we cannot know whether our data processing methods successfully removed the disturbances or if they just removed everything.

To have the data contain measurable brain activity that is unrelated to the head and eye movements, we used a so-called oddball paradigm. In an experiment that follows the oddball paradigm, a subject is provided with two types of stimulus: a standard stimulus that occurs more frequently and a target stimulus that is less frequently occurring. The subject needs to keep track of the less frequently occurring target stimulus. This way the target stimulus is known to cause a different amplitude in the brain response's P300 component when compared to the P300 caused by the standard stimulus [24]. The P300 component is a good choice for this purpose, because it has a relatively high amplitude and is therefore easier to recover compared to other components in the presence of noise. It is also a very well known component, and a lot of research has been done on it. Figure 4 shows what the P300 (a.k.a. P3) and earlier sensory components (P1, N1, P2, N2) should approximately look like.

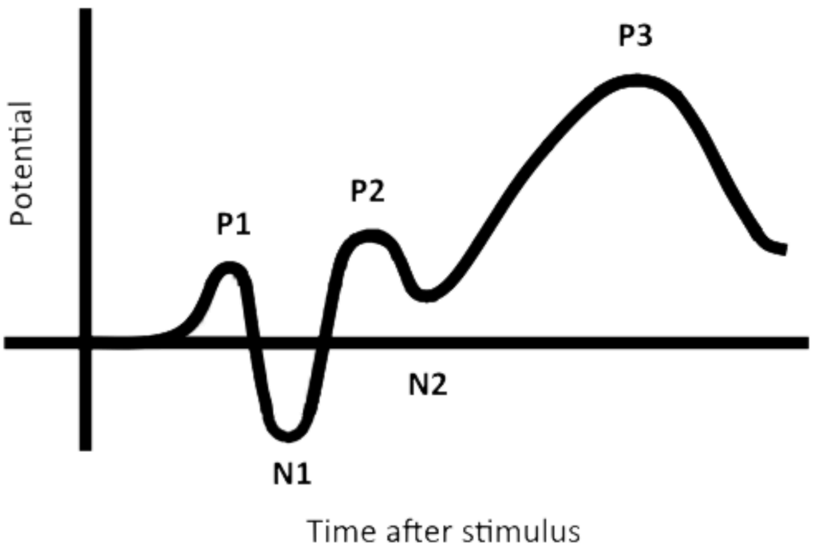

Figure 4. An illustration of what the ERP to the oddball stimulus should look like.

The P300 is an ERP component that is associated with tasks that require stimulus discrimination [25]. The name P300 comes from the fact that it's a positive potential that peaks somewhere around 300 milliseconds after the event. According to Polich [25], P300 can be subdivided into P3a and P3b components. The P3a component is essentially caused by something that happens as a total surprise to the subject. In an oddball experiment it could be some third type of stimulus that the subject was not informed of before starting the experiment. The P3b component on the other hand is caused by something that the subject knows is going to occur, but doesn't know when. It is also something that the subject is paying attention to. Therefore in our oddball experiment, P3b should be the response to the oddball stimulus.

Our ideal goal for this study would be to remove only the unrelated components from the data while preserving the brain activity. If our noise and artifact removal is successful, the P300 along with other ERP components should still be visible in the data after all the processing has been done.

The version of the oddball paradigm that we used consisted of auditory stimuli. The subject is provided with a series of audible beeps, some of them being low toned and others being high toned. The lower toned beeps occur with a higher frequency, and the subject is asked to count the less frequently occurring high toned oddball beeps.

For the implementation of the oddball experiment, we modified an already existing Python script that our supervisor provided for us. The Python script makes use of the Psychopy [26] package. The code sends a trigger to the TriggerBox each time there is a beep. A different trigger is used for the standard and oddball beeps to differentiate between them. This way we can separate the brain responses to the standard and oddball stimuli. The high tone beep should cause a significantly larger P300 response to appear after it's trigger mark than the lower toned beep. This P300 effect should be clearly visible. Even though the P300 response is present in both cases, it should be very much larger for the oddball beeps.

The experiment consisted of multiple blocks. Each block had a series of beeps, and the subject was asked to state the number of higher toned beeps they counted at the end of each block. Counting the beeps should motivate the subject to pay attention to them, causing a measurable P300 effect in the brain activity data. A total of 8 blocks were run, of which blocks 1, 3, 5 and 7 had the subject stay still and perform only the auditory oddball experiment, while the rest (blocks 2, 4, 6 and 8) also included the subject moving their head by tracking a video switching between 4 TV's in a VR environment. Each trial had 120 beeps, and the probability for a beep being an oddball was 25%. The VR environment is described in more detail in the following section.

### *3.1.3. Virtual Reality Environment*

For the VR part of the experiment, we created a VR environment using Unity game engine. It consists of a room where the subject is placed in the middle. There are televisions on each of the four corners of the wall in front of the subject. A video is playing on one of the TV's at any given time, and the TV in which the video is playing switches randomly. The subject is supposed to watch, or at least look at the video, so when the active TV is changed, the subject moves their head. In this way, we hoped to cause head and eye movements that are at least somewhat natural and unrelated to the auditory oddball experiment.

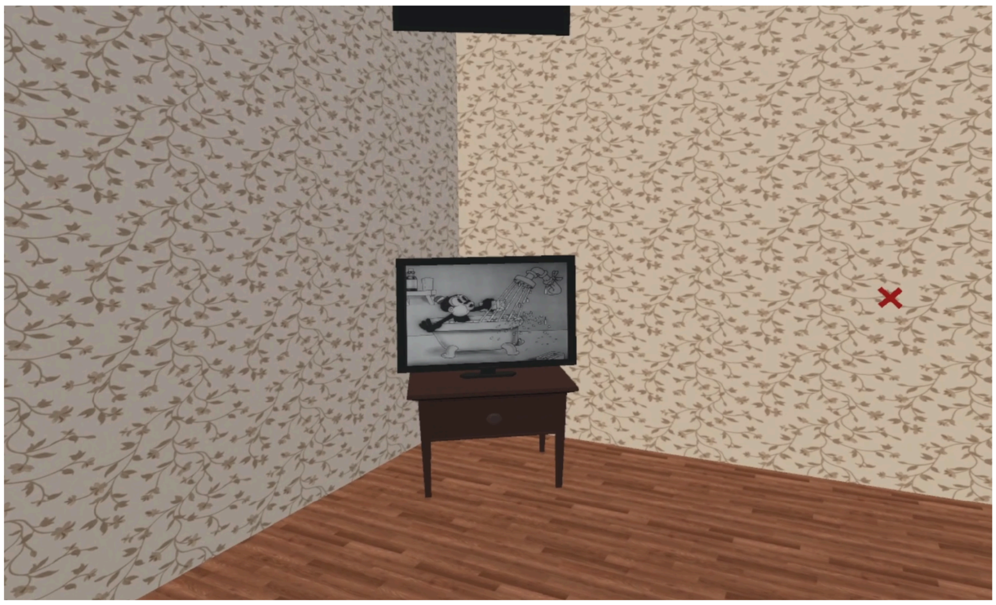

Figure 5. Subject facing the bottom left corner in the VR environment.

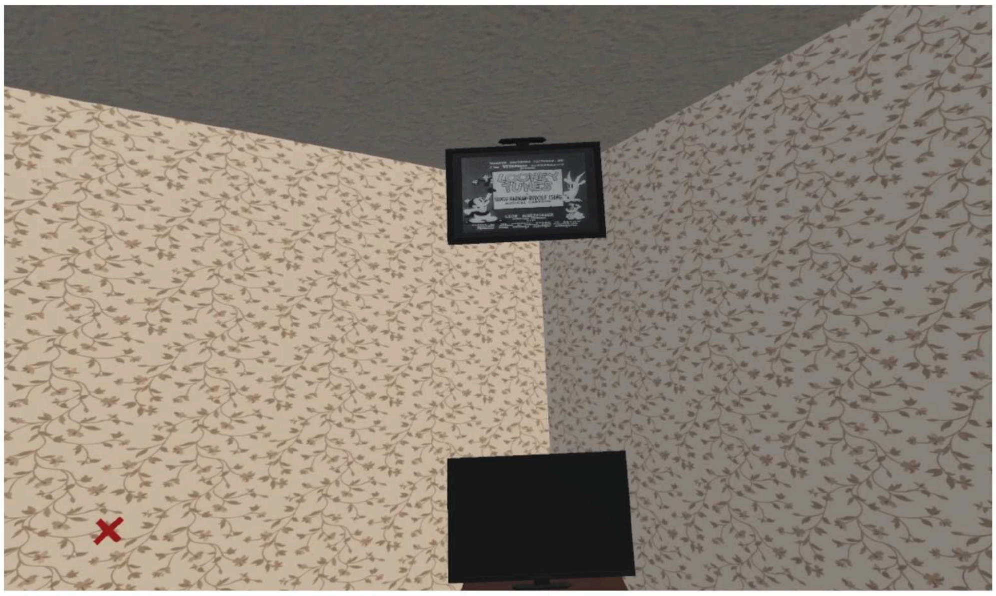

Figure 6. Subject facing the upper right corner in the VR environment.

The video played on the TVs is an old cartoon called "Sinkin' in the Bathtub" by Warner Bros. It was first aired in April 19th 1930 and it's copyright has already expired, so we could comfortably use it here. The television in which the cartoon is playing switches with an interval ranging from 4 to 8 seconds. The interval is chosen randomly every time after the active TV has switched. The television is also chosen randomly, except that the same television can't be chosen two times in a row. C# code that was used to implement this TV switching logic can be found in Appendix C.

We initially planned to create a more complex VR environment, where the subject would have been moving through a more diverse environment such as a virtual art gallery or a virtual university campus. The task in those cases would have been to just observe the surroundings where interesting objects would have been placed. However, we eventually ended up deciding to implement the TV setting. Our setting gives a more controlled environment while still leading the subject to move in ways that we are looking for. It also makes the head turns more similar with each other, which should make it easier for ICA to extract them as one source. A more complex VE also could have been a bit overwhelming for the subject, considering the simultaneous auditory oddball experiment. Our environment was also much more time efficient to implement while serving the same purpose as the other ideas.

The reasoning behind using TV screens is that we also wanted to prevent the subject from getting too bored. Boredom may have an effect on brain activity and therefore also on the ERP, so we wanted to try to prevent it. We hoped that this TV setting would be sufficient enough to keep a subject engaged.

The HMD we used was an Oculus Rift S. We were initially planning to use a Meta Quest 2. However, we encountered some significant problems with it. Random disconnections would occur and sometimes the device suddenly refused to show any picture. This is obviously unacceptable, since it can happen in the middle of an experiment. We managed to get our hands on the Rift S and we would recommend others to opt for it as well. The HMD was connected by a cable to a regular desktop computer running Windows 10, where the application was running.

### *3.1.4. Trigger Server*

In order to have the VR environment and the oddball experiment software communicate with the TriggerBox, we had to create an additional small piece of software that we call the trigger server. The trigger server is a small piece of Python code that opens a UDP socket and receives trigger commands as UDP packets. A single packet contains an integer that the trigger server will then forward to the trigger box via a USB virtual serial port. The VR environment sends triggers when the subject turns their head and the oddball experiment sends one when a beep occurs.

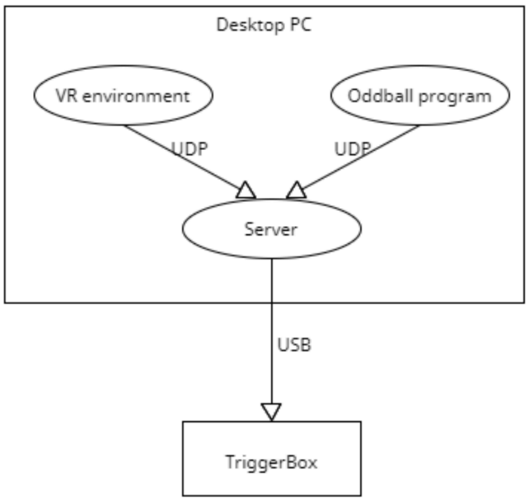

Figure 7. Trigger server setup for the experiment.

The reason for implementing this was that we found there to be a limitation in the Windows operating system that prevents two programs from using the same port simultaneously. This is a problem because both the VR environment and the oddball experiment need to send triggers to the TriggerBox. With this small piece of software, we were able to make our experiment setup work without having to order more equipment. The code of this trigger server can be found in Appendix A.

### *3.1.5. Head Movement Detection*

To mark the times when the subject was turning their head, we implemented a head movement detection feature in the VR application. The reason we did this was that it would be helpful during the data-analysis phase to know when the subject was turning their head. We could compare the scroll data before and after the processing phase and see if the disturbances around the head motion trigger marks are removed or attenuated.

Since we were using Unity for the VR environment, the head movement detection was simply implemented as part of the VR application. The head movement detection detects the orientation of the subjects head by reading the orientation of the camera object in the Unity VR environment. The camera object corresponds to the VR headset, and therefore it turns as the subject turns their head. Practically speaking, the movement detection works by reading the orientation of the camera object every 50 milliseconds, calculating the deltas (differences between two successive readings) for the previous 20 readings and then calculating the average of those deltas. This is done separately for the three axes that the camera object can rotate along. If the average delta along any axis is greater than a certain threshold, then a head movement is detected to have begun. When the average delta along all three axis drops below a certain threshold, then the head movement is detected to have ended. Suitable threshold values were found by experimentation. The C# code that implements the head movement detection and sending the triggers to the trigger server can be found in Appendix B.

#### *3.1.6. Experiment Session*

During the experiment, a subject is performing the oddball task while also looking at the video playing on one of the VE TV's, with said TV changing randomly. At the same time, they should keep track of the higher toned beeps. As mentioned before, the VR environment and the oddball experiment were not synchronized in any way.

For testing purposes, we first conducted a pilot experiment with a simpler VR environment where we had just two TVs. It confirmed that our setup was working properly but we wanted to expand to using four TVs in order to have more variety to the motions of the subject's head.

### 3.2. Data Processing

After the data collection, the data processing phase can begin. In simplified terms, the data processing includes applying ICA in combination with filtering, to remove the unwanted artifacts and noise, such as line noise caused by electrical outlets in the lab. After processing the data with the appropriate methods, we should be able to see whether the artifacts and noise were removed and if the brain activity was preserved.

#### *3.2.1. Tools*

Our preferred method to post-process the data was by using Matlab. Matlab has a great toolbox called EEGLAB [27] that makes dealing with EEG data easier compared to generic signal processing tools. With this toolbox, we also used a plugin called ERPLAB Toolbox [28] that gives some important additional and improved features, such as event lists.

There are many ways to run ICA in Matlab, notably RUNICA and AMICA. Both of them are different implementations for ICA. RUNICA is included in the EEGLAB Toolbox which makes it a very accessible choice. AMICA however, needs to be downloaded separately but it integrates into EEGLAB seamlessly after acquiring the required files.

Delorme et. al. compared the performance of different ICA algorithms in separating different sources from EEG data, and found out that AMICA is generally the best performing algorithm available [29]. In addition, judging by the results from our pilot runs, it seemed to do a better job of removing blinks compared to RUNICA. Therefore we decided to run AMICA. However, it can relatively slow, so we would recommend using a relatively powerful PC for it in order to make the process less painful.

### *3.2.2. Data Processing Pipeline*

In detail, our way of processing the data went as follows.

1. Downsample the data to 250 Hz
2. Edit channel locations to match the used 10-20 system
3. Aggressively high-pass filter ($f_c$ = 1 Hz)
4. Remove miscellaneous artifacts from the data
5. Run AMICA
6. Run a correlation script to find a relationship between the ICA components and head turns
7. Give the ICA weights to the dataset constructed prior to the aggressive filtering
8. Label ICA components with ICLabel
9. Inspect the ICA components and remove eye and head related components based on the correlation script's results and ICLabel
10. Bandpass filter (0.1 Hz - 30 Hz)
11. Re-reference to M1 and M2
12. Create an event list, bin epochs and extract them
13. Average the data

First of all, we want to downsample the data in order to compress the data size which can help speed up the processing. As mentioned earlier, our amplifier collected data at 1000 Hz. Chapter 16 in Steven Luck's book suggests that 250 Hz is a suitable sample rate for most EEG experiments [3].

In turn, high-pass filtering aggressively is mainly for removing baseline drift. The high pass filter that was used was an IIR Butterworth filter with a half-amplitude (-6dB) cutoff at 1Hz frequency and an order of 2.

Removing miscellaneous artifacts from the data is probably the most eye catching step here. Steve J. Luck calls these artifacts "crap" or “commonly recorded artifactual potentials" [3]. This can be done through ERPLAB's artifact rejection with the following settings.

- 250 uV threshold
- 500 ms window
- 250 ms step
- Pre-filter from 20 to 120 Hz
- Connect segments of less than 1000 ms gap
- Give a 100 ms buffer

After AMICA, we used a custom script that looks through each of the components and tries to find a correlation between the components and head motions. The script generates a vector that contains zeros for the length of the EEG data except the times between head motions. The periods between head motion trigger marks are filled with ones. The script then finds correlation values for each of the ICA components compared to the vector. The source code for the script can be found in Appendix D.

After ICA is done, you should pass the weights to a dataset constructed prior to the aggressive filtering and proceed with that. Otherwise you will have a dataset with 1 Hz high-pass and chunks removed.

Usually, ICLabel in EEGLAB does a good job of labeling eye related components. Still, you should use caution when observing the generated labels. In the case of head motions, there are no labels for them in the standard EEGLAB labeling.

After finally removing the components that were determined as unwanted, we band-pass filter the data. This is to filter out high and low frequency oscillations that do not necessarily have much to do with cognitive processes. The filter used for this was an IIR Butterworth filter with a half-amplitude (-6 dB) lower cutoff at 0.1 Hz frequency and a half-amplitude (-6 dB) higher cutoff at 30 Hz frequency, and an order of 2.

Then we bin and extract epochs based on the events that we have in the data. In our case, they are the standard stimulus and oddball stimulus, which are the events that the data should contain brain responses to.

We used the Cz electrode as our reference during the data collection. This was to make sure that head motion related components are well included in the data. The assumption is that those components come from the electrodes near the neck so we used a reference electrode that is located near the top of the scalp. For the purpose of observing the ERPs in a graph, we re-referenced the data to the two mastoid electrodes (M1 and M2). This is the most commonly used reference in ERP research done on cognition, which makes our results easier to compare to the results of others.

The implementation and results of this data processing pipeline are explained in more detail in the later data processing chapter.

# 4. EXPERIMENT IMPLEMENTATION

The design section covered the design and reasoning behind the our study. In this chapter, we explain the conducted experiment session during which the data collection happened.

## 4.1. Experiment

### *4.1.1. Participants*

Since our study is rather experimental and demonstrative, we did not feel the need for having a large participant pool. Motivating people to participate in our study during a limited time period also would have been rather difficult, especially considering the relatively complex experiment type and long setup times. The scope of a bachelor's thesis also does not allow for an extensive study and we were using a shared laboratory space.

We ran one comprehensive session, in which we collected all the necessary data for our purposes. The sole participant was a 31-year-old male who had previous experience with VR and EEG. This was great for our purpose since the technology used was not a distracting factor and the subject felt comfortable.

### *4.1.2. Procedure*

The subject was seated in a chair while they wore an EEG cap under an Oculus Rift S. In front of them was the PC running the software of the experiment which we controlled.

First, the subject was explained the setup procedure. They had a chance to use the restroom before the electrode cap setup. Everyone in the room made sure their phones were on silent to not disturb the experiment.

The subject was instructed to comb or brush their hair which is helpful when it comes to getting a proper connection between the scalp and the electrodes. The subject's head was measured in order to determine the correct cap size. To prepare for the face electrodes, the electrode locations were cleaned with NuPrep gel and alcohol pads. A syringe with a dull needle was used to place gel to make a connection between the scalp and the electrodes. The needle is also helpful for spreading hair to see the scalp though each electrode hole.

Once all the electrodes were on the cap and the impedances showed below 10 kOhm, the HMD was carefully placed on the head. No disturbed impedances appeared during this process so we were ready to proceed.

Before wearing the HMD, the subject was explained their tasks. They were asked to listen to the beeps and count the amount of high pitch beeps that they hear. At the end of a block, they would then be asked for the amount they managed to count.

During every other block, they were also instructed to follow the cartoon that was being played on the TVs inside the VE. During the other blocks, they could just stare at a motionless red X that was visible on the same wall with the TVs.

After the experiment, the subject was given the ability to wash their hair in the bathroom provided with shampoo and a clean towel.

## 4.2. Outcome

Overall, the experiment was carried out successfully. We observed the subject's actions throughout the session and they seemed to do extremely well with following the given instructions.

### *4.2.1. Beep Counting*

The subject managed to stay focused and counted the high tone oddball beeps quite accurately, as shown in the table below. There was some concern that the experiment task could be a bit too demanding considering having two tasks at the same time. However, judging by these results, it was not a huge issue.

During blocks 1, 3, 5 and 7 the oddball experiment was used but the subject was asked to not track the TV's. In turn, during blocks 2, 4, 6 and 8, the oddball experiment was used but the subject was also asked to track the TV's. Even though the subject was able to count the beeps with good precision even while tracking the TV's, they always missed some with the simultaneous tasks. Overall the accuracy seems great and we could determine the experiment successful.

The accuracy shown in the table was calculated with the formula (120 - abs(subjects_answer - real_answer)) / 120.

| Block # | Subject's answer | Real answer | Accuracy % |
|---|---|---|---|
| 1 | 23 | 23 | 100% |
| 2 | 27 | 28 | 99.2% |
| 3 | 30 | 30 | 100% |
| 4 | 26 | 29 | 97.5% |
| 5 | 35 | 35 | 100% |
| 6 | 33 | 32 | 99.2% |
| 7 | 34 | 35 | 99.2% |
| 8 | 32 | 36 | 96.7% |

Table 1. The amount of high tone beeps counted by the subject compared to the real value (Odd block no. = without head turning; Even block no. = with head turning).

# 5. DATA PROCESSING

## 5.1. Overview of the Data

By manually examining the data, we could determine that it was sufficient for our needs. We managed to capture a decent amount of artifacts and noise.

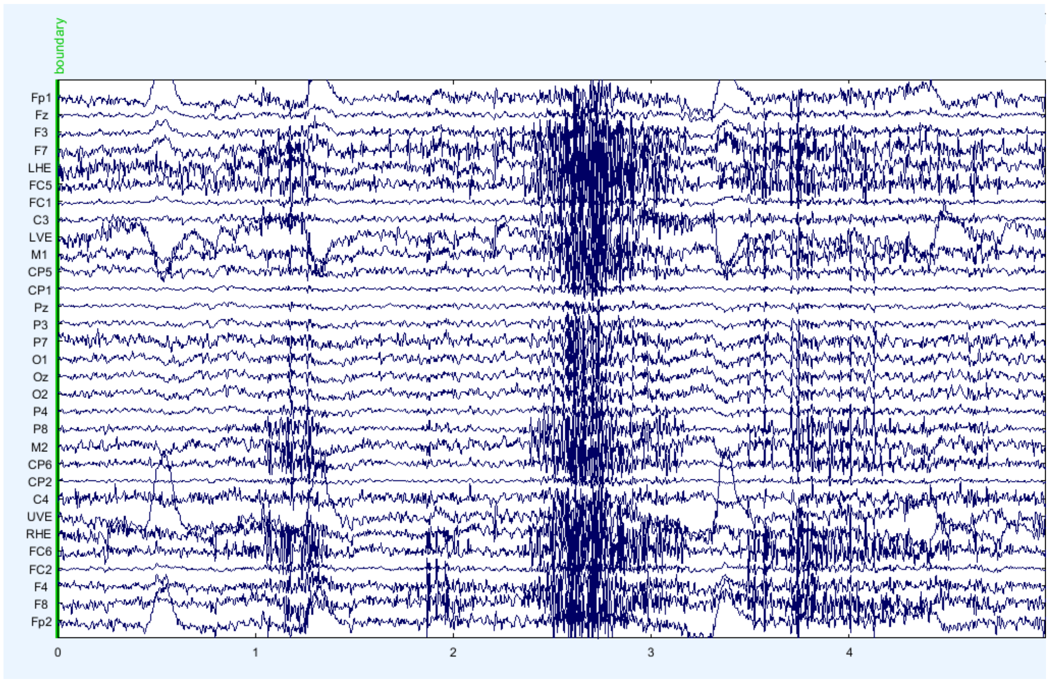

Figure 8. Significant artifacts when the subject is moving their head freely before the experiment.

Figure 8 shows an extreme example of noticeable artifacts. This was right after starting the recording but before starting the experiment. The subject was still moving their head around freely and clenching their jaw. This type of noisy data is also present during the breaks between blocks. As explained in the design section, step 4 of the data processing pipeline attempts to remove these types of artifacts before running AMICA because do not want the ICA algorithm to focus on them.

During the actual experiment, the head motion of the subject was more limited. During the blocks with the TV task, major head movement occurred only when the subject switched to look at a different TV. This can be seen in Figure 9, where the disturbance is between the head turn trigger marks S5 and S6.

As seen in the figures, head motion can indeed be problematic when it comes to getting clean EEG data. However, Figure 10 demonstrates that even if the subject is holding their head still, the data still gets contaminated with artifacts. In this example, we can see two noticeable blinks during a block when the subject was only facing towards the motionless X on the wall.

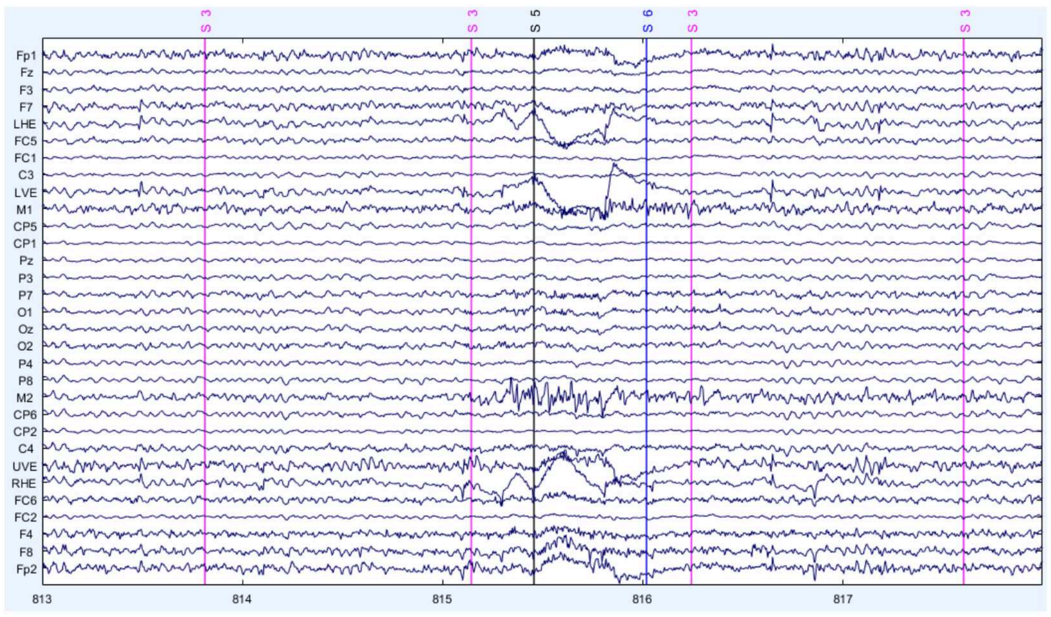

Figure 9. Head turn during the actual experiment causing disturbance in the data.

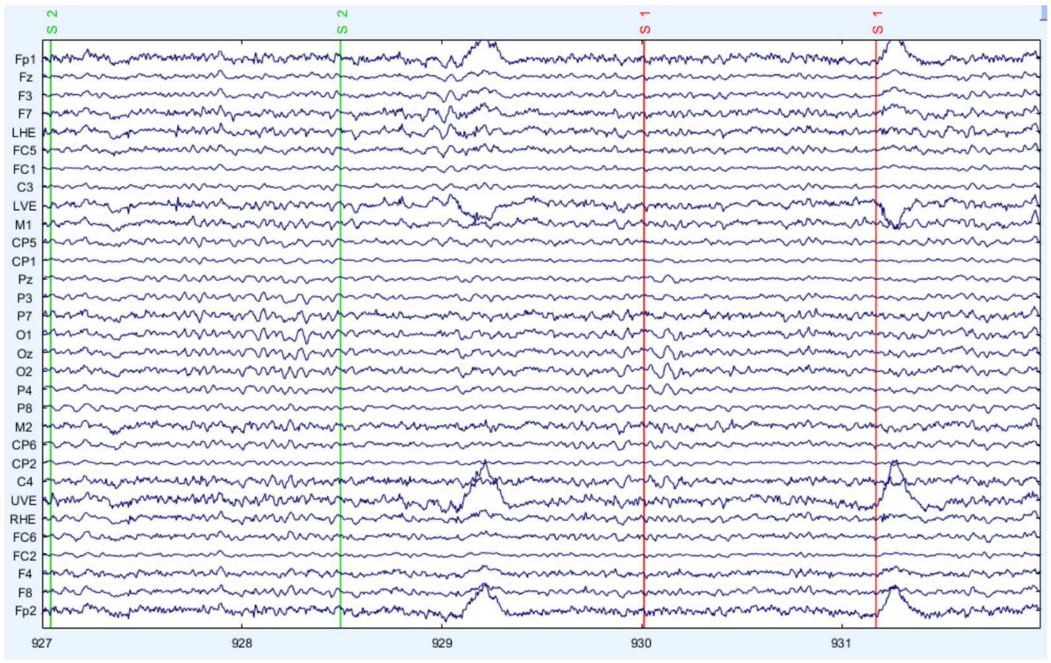

Figure 10. Two eye blinks causing prominent spikes in the data.

#### *5.1.1. Trigger Marks*

As discussed in the design section, we used trigger marks to mark interesting events in our data. These trigger marks can be seen in many figures throughout this chapter. The following table explains the meanings behind the different labels for the marks.

| Label | Meaning |
|---|---|
| S 1 | Standard stimulus without TV task |
| S 2 | Oddball stimulus without TV task |
| S 3 | Standard stimulus with TV task |
| S 4 | Oddball stimulus with TV task |
| S 5 | Head motion start |
| S 6 | Head motion end |

Table 2. Trigger mark labels explained.

### 5.2. Baseline - Removing Only Eye Related Artifacts

Our goal was to see if we can remove head movement related artifacts from the data using ICA. However, in order to analyze our results, we first needed some baseline data. As discussed before, ICA has been found to be effective and useful especially with ocular artifact removal [7]. Therefore for the baseline, we processed the data with this more of a standard approach. Here we only remove ICA components related to eye movements and blinks, in addition to other common noise sources, such as line noise.

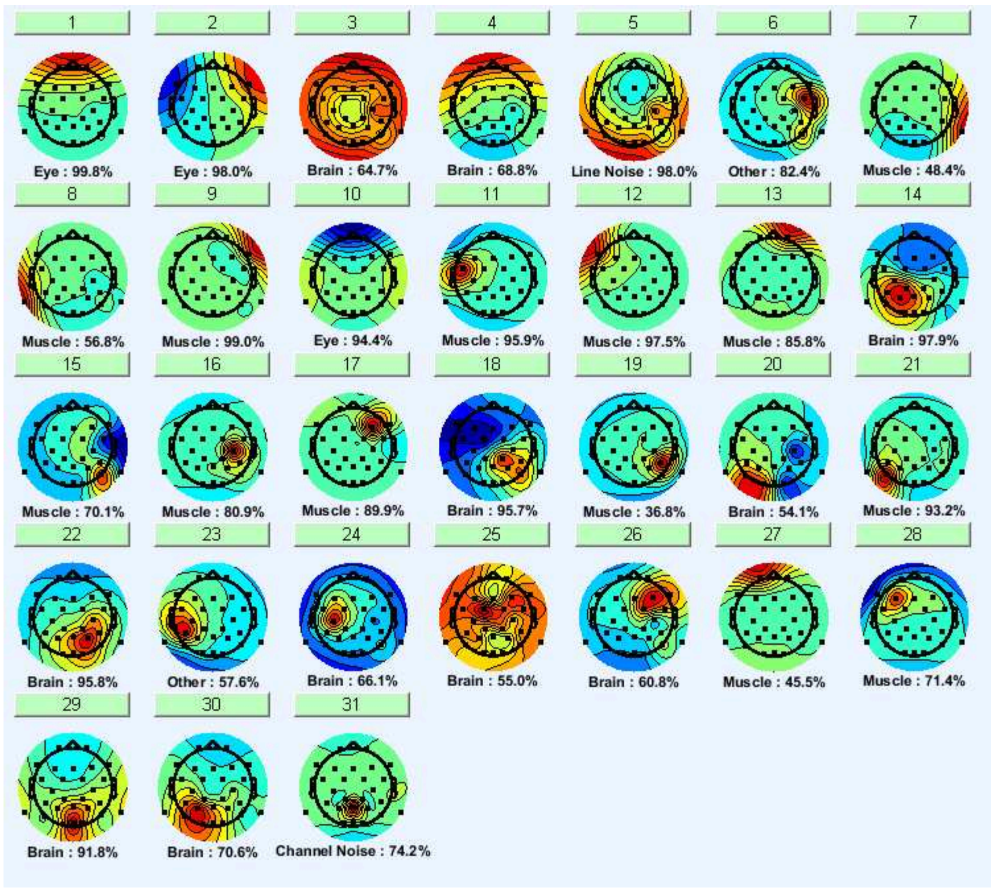

Figure 11. ICA components separated by AMICA and labelled by ICLabel.

Figure 11 shows the ICA components that AMICA was able to separate. The components have been labelled with ICLabel. Starting from 1 and counting towards 31, each component accounts for a smaller portion of the total electrical activity. Or in other words, the closer a component is to 1, the more significant it is, while the components at the end of the list are relatively insignificant.

Manual inspection of the components in addition to the labels is necessary to make a reasonable decision on which components to remove.

Components 1 and 2 are labeled by ICLabel as eye related with nearly 100% probability, and their scalp distribution also suggests that they are related to eye movements. These two can be confidently removed.

Component 5 was identified as "line noise" by ICLabel. Also, from Figure 12 it can be seen to contain a huge spike at 50 Hz. This suggests that the component very probably is noise from the AC outlets in the room, so it was also marked to be removed.

Component 10 also looks to be heavily eye movement related, similar to components 1 and 2. This could also be removed without a reason for doubt.

Finally, the activity in component 31 suggests to be coming from a bad electrode, so it was also removed. We suspect that the headband of the VR headset might have been rubbing against this electrode. It was labeled as "channel noise" by ICLabel. And as Figure 13 shows, it also has a prominent spike at 50 Hz, so it can also be marked for removal.

So in total, components 1, 2, 5, 10 and 31 were removed.

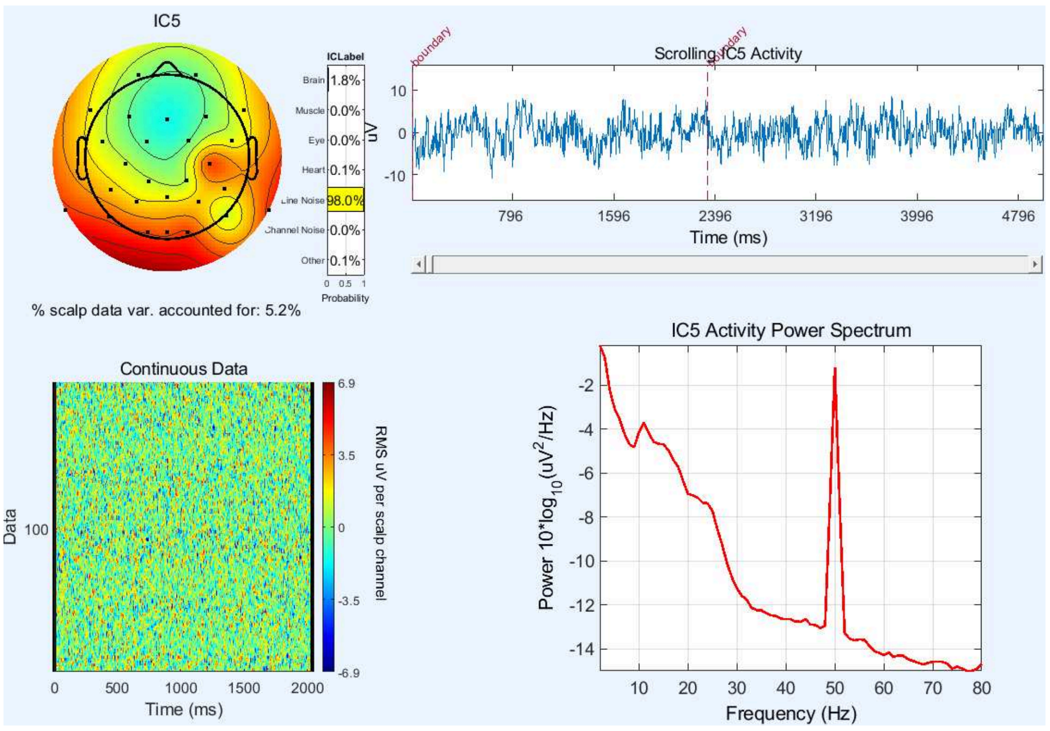

Figure 12. Closer view of component 5.

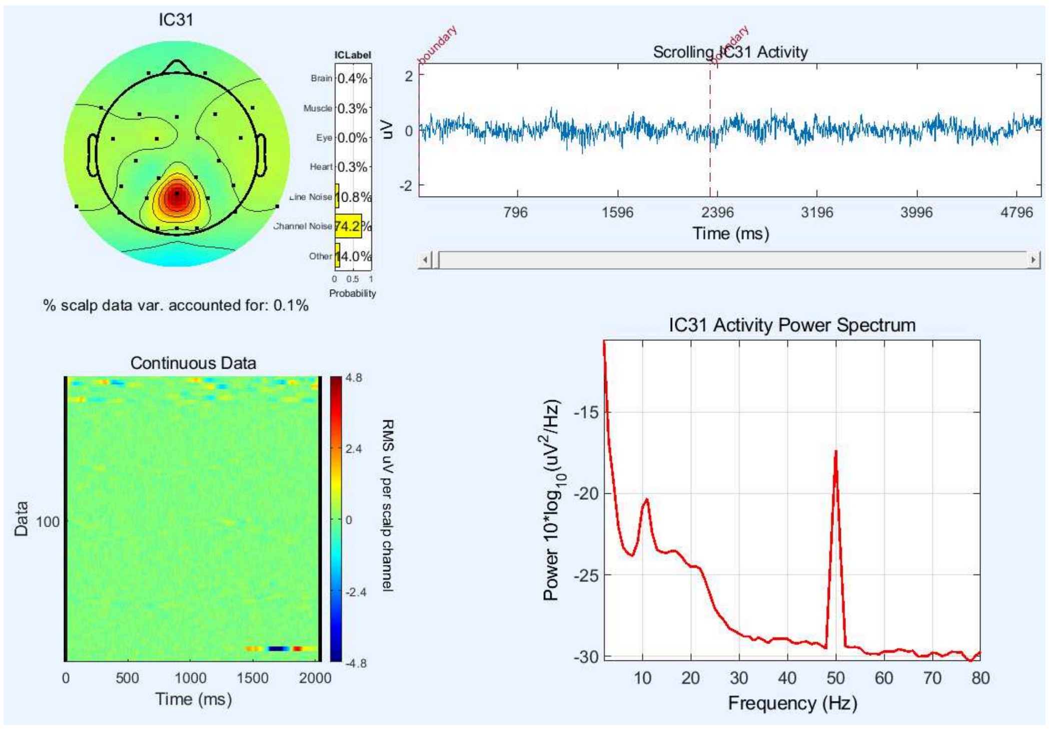

Figure 13. Closer view of component 31.

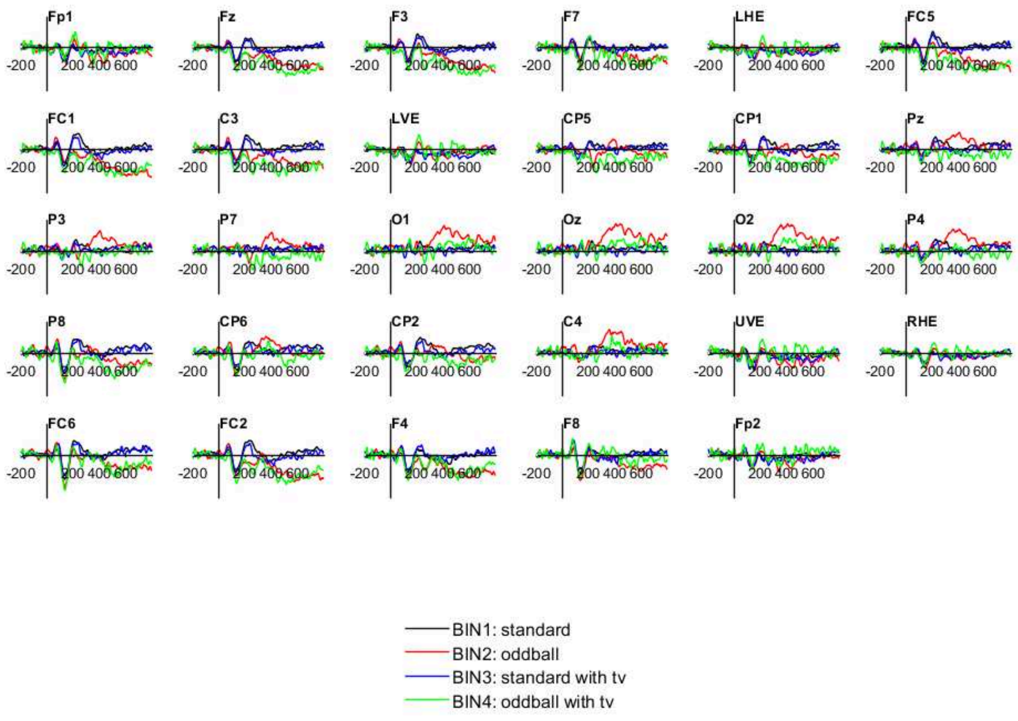

Figure 14. ERP waveforms for different channels.

After removing the unwanted components, we proceeded with the rest of our data processing pipeline.

Figure 14 shows the resulting ERP waveforms for different channels. However, in order to properly examine the ERP components, we want to focus on an individual channel. Our original intention was to focus on Pz because P300 component is strongest in the parietal lobe region (see Figure 2). However, based on the ICA component 31 (see Figure 11), it looks like the Pz electrode was interfered by the headband of the HMD, as we already discussed earlier. Therefore we had to use some other electrode instead. P4 is relatively close to the Pz location, so we decided to use that one.

By examining Figure 15, we can see that it mostly matches the theoretical model (see Figure 4). However, we should see two P300 effects. Only the P300 for the oddball without the TV's is visible. For the oddball with TV's we can see that the P300 component in the ERP is quite nonexistent. This is a very surprising result. However, it could be that we have not removed the head movements components yet. We discuss this more in the discussion chapter.

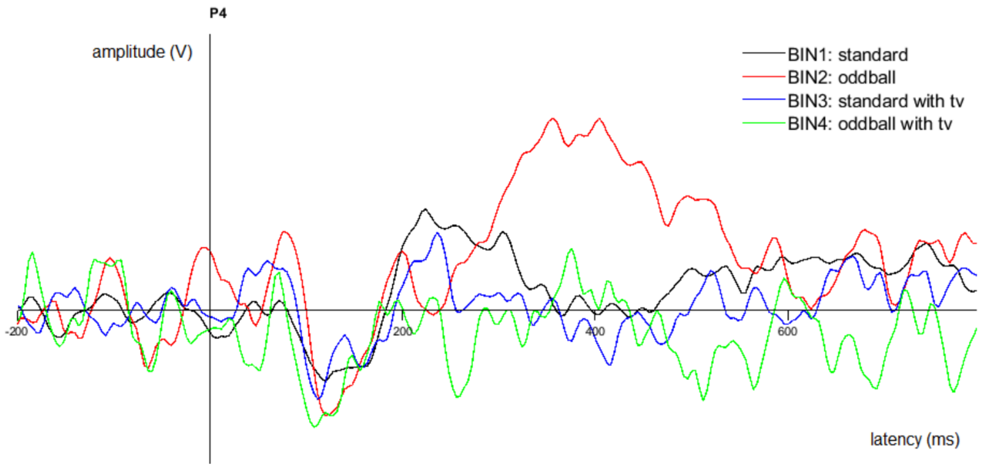

Figure 15. ERP waveform for P4 channel.

By examining the scroll data, we can determine that our eye component removal was successful. Figure 16 shows that the spikes caused by blinks seen in Figure 10 have now disappeared.

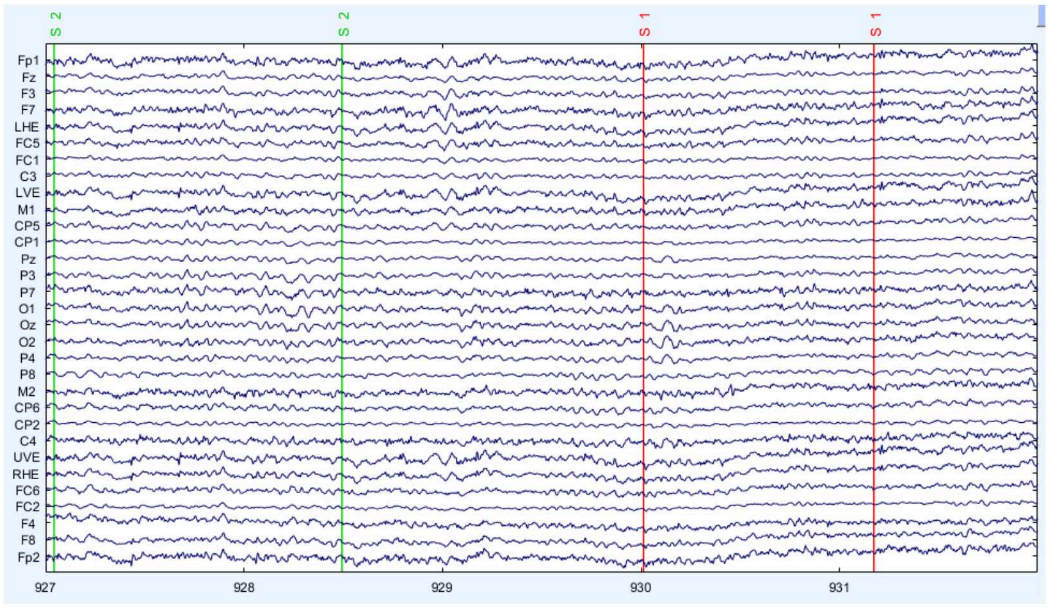

Figure 16. Spikes caused by blinks have been removed or significantly attenuated (compared to Figure 10).

### 5.3. Removing Head Motion Related Artifacts - - Conservative Approach

In the previous section, we only removed ICA components that seemed to be related to eye movements, and also some other noisy components, to establish a baseline. Now, we move on to the topic of this paper, removal of head movement related components.

First, we want to remove the same components as we did for the baseline. Those include components 1, 2, 5, 10 and 31.

In addition, we want to remove the head motion components. To help with the decision of which ones to remove, we used the earlier mentioned script (see Appendix D) that finds the correlation between each of the ICA components with the head turn markers. The script should be run before giving the ICA weights back to the dataset constructed before aggressive filtering.

Table 3 shows the correlations of each ICA component calculated by the correlation script. Components where correlation was greater than or equal to 0.100 were considered to be potentially related to head movements, but manual inspection was conducted for confirmation.

| ICA Component number | correlation |
|---|---|
| 1 | 0.086 |
| 2 | 0.245 |
| 3 | 0.008 |
| 4 | 0.118 |
| 5 | 0.039 |
| 6 | 0.014 |
| 7 | 0.160 |
| 8 | 0.118 |
| 9 | 0.056 |
| 10 | 0.090 |
| 11 | 0.020 |
| 12 | 0.036 |
| 13 | 0.079 |
| 14 | 0.024 |
| 15 | 0.001 |
| 16 | 0.068 |
| 17 | 0.056 |
| 18 | 0.022 |
| 19 | 0.003 |
| 20 | 0.063 |
| 21 | 0.079 |
| 22 | 0.000 |
| 23 | 0.016 |
| 24 | 0.008 |
| 25 | 0.142 |
| 26 | 0.009 |
| 27 | 0.044 |
| 28 | 0.043 |
| 29 | 0.013 |
| 30 | 0.009 |
| 31 | 0.064 |

Table 3. Correlation of each ICA component with head turn markers.

For the head motions, we also want to consider the labels produced by ICLabel. ICLabel tends to do a good job at labeling eye related components and often other noise sources, as could be seen in the baseline section. However, ICLabel has not been trained to separate head movement related components so the labels should be considered as indicative guesses. Some head movement related components might be labeled as "muscle", but again, we cannot blindly trust the labels.

Judging by the correlation value, component 4 correlates with the head turns, but because of the topography and temporal distribution, we had suspicions that it might contain a lot of brain activity as well, so it was not removed. Components 7 and 8 also correlated with head turns. But in addition, their scalp distribution also suggests that they could have something to do with head movements, so they were also removed.

Component 25 has a strange looking scalp distribution and it also seems to correlate with head movements, so it was removed. ICLabel thinks that it is brain related, so it is also possible that it includes some brain activity related to the head movement. However, we decided that it is reasonably safe to remove because ICA components this far down the list do not have a very significant impact anyway.

All in all, components 1, 2, 5, 7, 8, 10, 25 and 31 were removed.

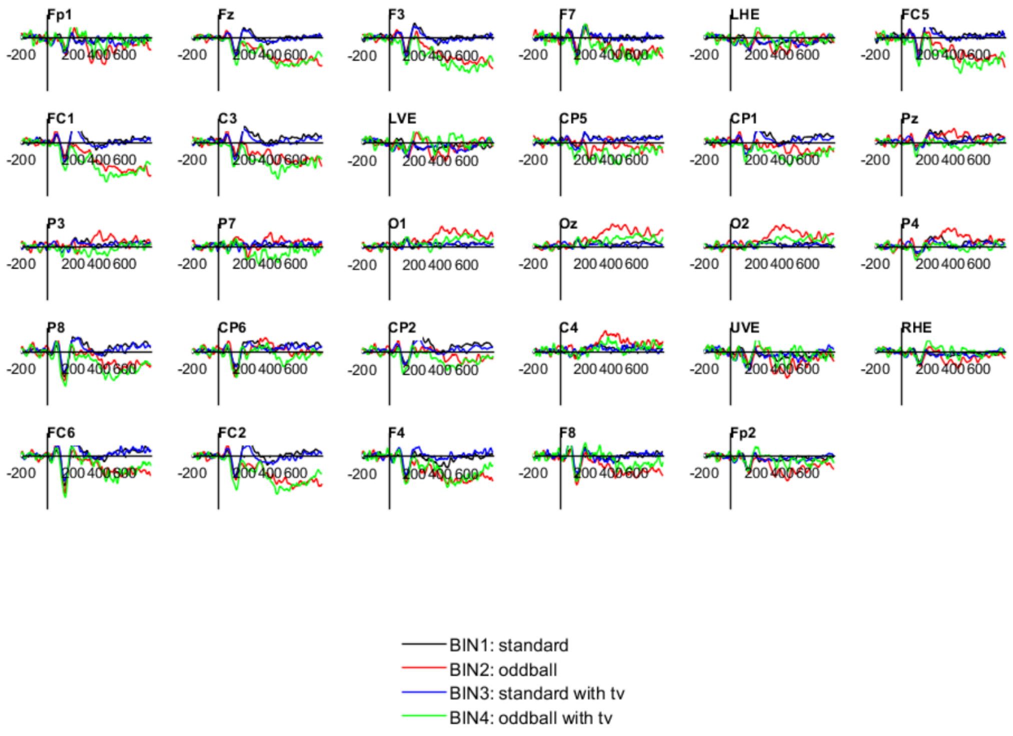

Figure 17. ERP waveforms for different channels.

Figure 17 shows the obtained ERP waveforms for different channels. Figure 18 shows the ERPs in the P4 channel. Only one P300 (for the oddball without TVs) can been seen, which is expected based on the results from the baseline.

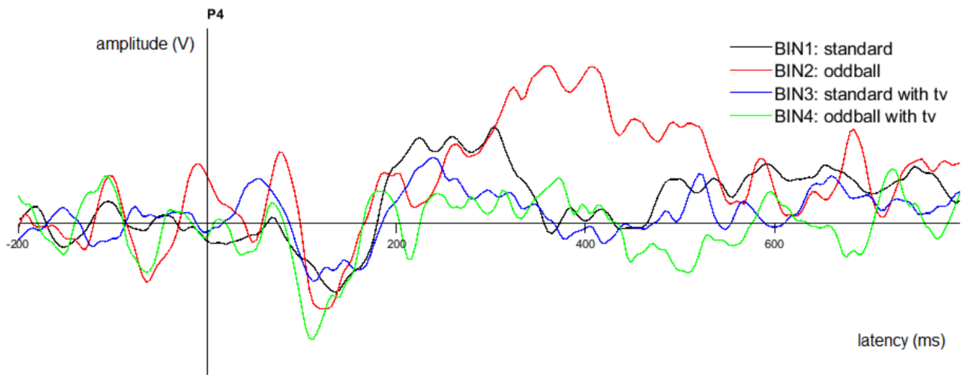

Figure 18. ERP waveform for P4 channel.

In the baseline section we saw the two blinks disappear from the scroll data after successful processing. Now, we can see similar progress with head turns. In Figure 19, the head motion related disturbances between the head motion trigger marks have significantly attenuated compared to Figure 9. However, there are still parts of them left.

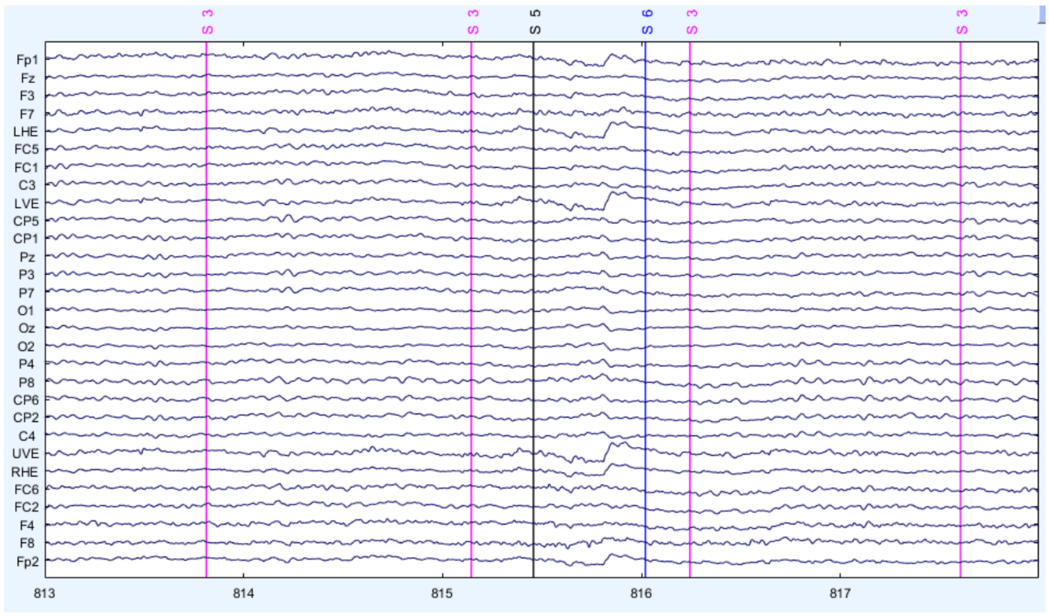

Figure 19. Disturbances have attenuated (compared to Figure 9).

### 5.4. Removing Head Motion Related Artifacts - Aggressive Approach

Our initial attempt at removing head movement artifacts from the data was somewhat inconclusive. For example, judging by Figure 19, there are still some clearly visible disturbances in the data, even if lower in amplitude compared to our baseline. This second approach is just a more aggressive version of our first attempt.

We again removed components 1, 2, 5, 7, 8, 10, 25 and 31 from the data, with the same reasoning as before. However, this time we also removed component 4. As was previously found out, it correlates with the head movements quite a lot, but we suspected that it might contain a lot of brain activity as well, and therefore we did not remove it in the first attempt.

So in total, components 1, 2, 4, 5, 7, 8, 10, 25 and 31 were removed this time. Figure 20 contains the ERP waveforms again. Figure 21 on the other hand contains the P4 channel. The resulting ERP waveforms only have barely noticeable differences compared to the previous ones. We expected the removal of component 4 to have a bigger effect.

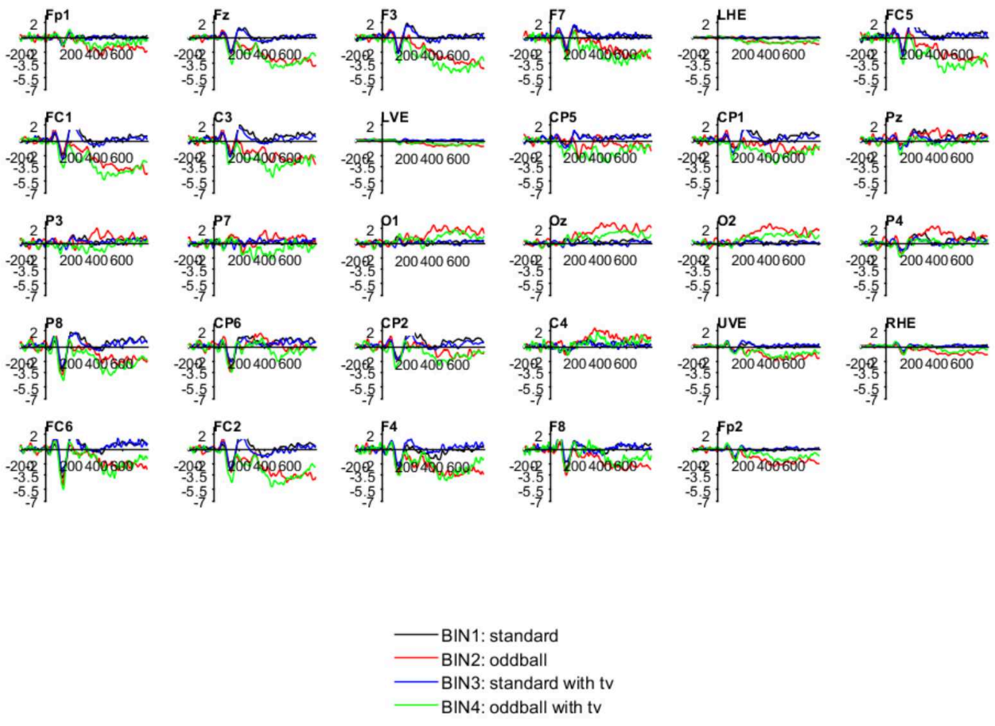

Figure 20. ERP waveforms for different channels.

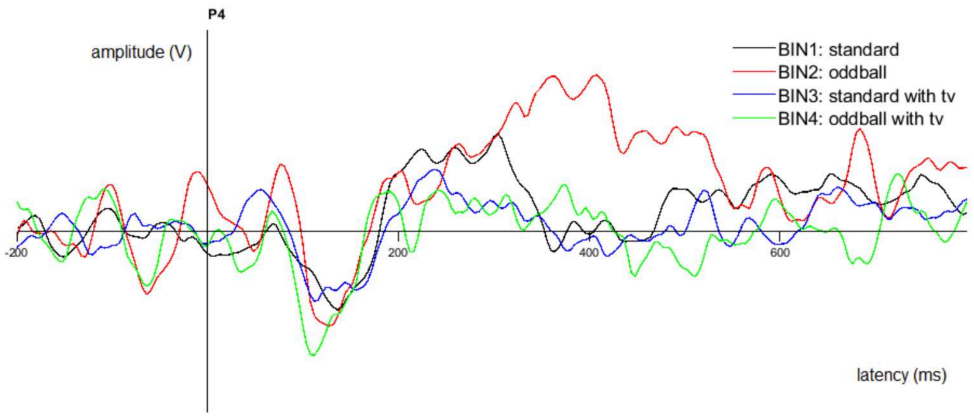

Figure 21. ERP waveform for P4 channel.

However, the scroll data seen in Figure 22 shows that the disturbances between the two trigger marks have now almost fully disappeared. The component 4 indeed did include head movement artifacts, but the impact of it in the ERPs is still quite insignificant.

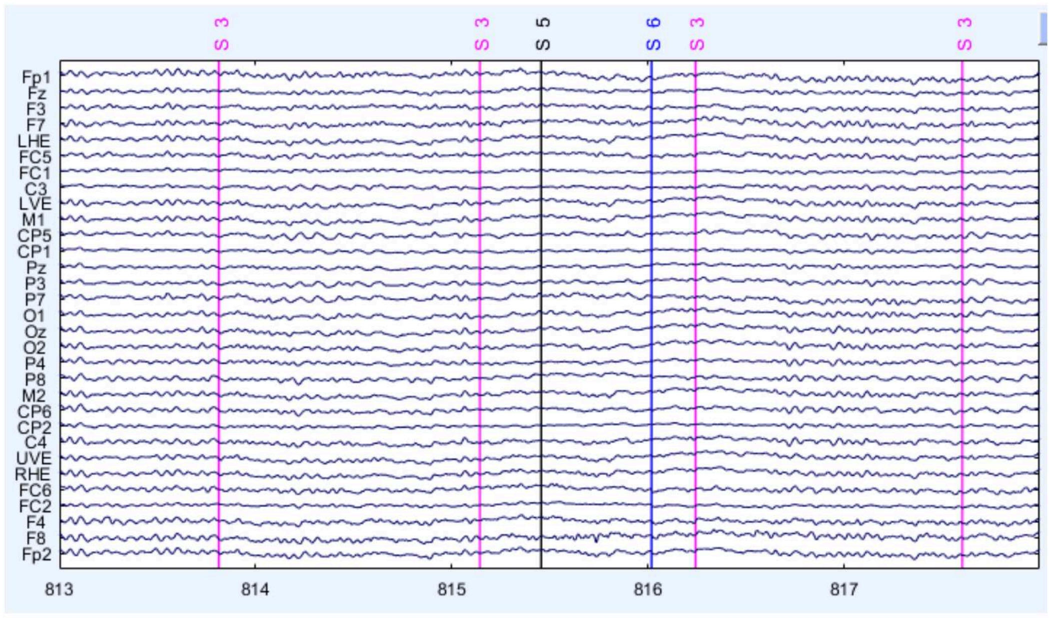

Figure 22. Disturbances have been removed or significantly attenuated (compared to Figure 9 and Figure 19).

# 6. DISCUSSION

## 6.1. Result Analysis

Our goal was to use ICA to remove head movement related artifacts from the EEG data, while not affecting ERP components of interest, such as the P300.

Already during the baseline processing phase, it could be seen that the P300 is nearly nonexistent for the oddball with head turns. This was a bit surprising to us since we were expecting clear P300s for both oddball stimuli.

There are a couple of reasons we think could be behind the missing P300. The TVs randomly switching could cause a P300 component of its own. The different components could then get mixed up and attenuate each other, or cause some other effect to make the component disappear from the ERP. Further research would be needed to more confidentially tell whether this is true or not.

Already during the design phase, we had a concern that the experiment could be very demanding for the subject during the blocks with active TVs. The subject had to concentrate on the audio stimuli while following the visual cues at the same time. And as mentioned in the related work chapter, Ochoa and Polich discussed in their paper that asking subjects not to blink results in a decreased P300 amplitude [14]. The tasks used in our experiment may have had the same effect. This is also backed by the fact that the less demanding blocks of the experiment did have a clearly visible P300. However, the subject was able to count the oddball beeps with relatively high accuracy even in the more demanding blocks, but it is still possible that tracking the TV's takes too much focus away from the oddball task.

However, the missing P300 is not really an issue in the context of this paper. We only initially chose P300 as our point of interest due to its high amplitude compared to other ERP components. As we can see from the graphs, other components, such as N1, are present in each ERP. This suggests that the brain data was not removed with our processing methods. This also leads us to suppose that the missing P300 was not there in the first place, so the problem was with our experiment design rather than data processing.

We could determine that ICA was indeed able to distinguish head movement related components that we then were able to remove. However, the significance of the removal of them ended up very underwhelming. In Figure 23, we can see that the wave forms are not very different from each other. The ERPs for the head movement block's events did lose some fluctuation due to the additional component removal, but for example the N1 component is clear regardless of the methods used. And if we compare our conservative approach with the more aggressive one, we can see that they produced essentially identical results, even though manual examination of the scroll data showed that the aggressive method removed a bigger chunk of the examined artifacts.

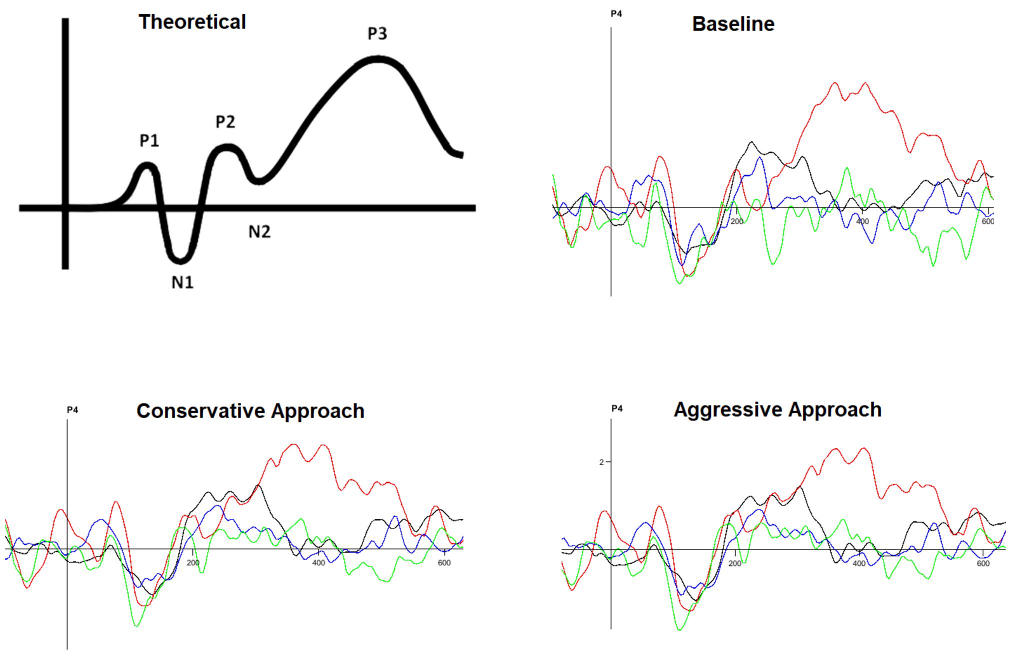

Figure 23. Comparison of the ERPs.

As mentioned in previous chapters, ICA should not be expected to get a perfect separation between components, so there is always a chance that there will be some brain activity that is also being removed. The P300 that we were able to recover is from the blocks during which the subject kept their head still, but the component still changes shape from the head movement component removal. This could confirm the idea of ICA's imperfect separation.

These results could also indicate that a big portion of the head movement artifacts could to be tangled with the eye components. The difference between the baseline and the two approaches are not significantly different, so maybe the "eye related components" found during the baseline creation already contained a big portion of the head movement related artifacts as well. This also backs the idea that people tend to move their eyes a lot while moving their head.

All in all, it appears that removing head movement artifacts might not be a huge concern even while utilizing VR. By limiting the amount of subject's head movements to not occur continuous throughout the experiment might be an effective way to keep the ERPs clean. In addition to that, traditional methods for ERP research, such as averaging, should take care of the outliers anyway, especially with a large participant pool.

## 6.2. Limitations and Concerns

In this section we discuss the factors to consider when interpreting our results.

#### *6.2.1. Experiment Design*

Obviously, one of the main limitations was also the extremely small amount of subjects. With a large sample size, we could get more comprehensive results and maybe even see the missing P300 that did not happen to appear from our only subject's brain response. Or alternatively, we could have a stronger reason to believe that tracking the TVs is the cause for the nonexistent P300 or its lower amplitude.

We were initially planning to carry out the experiment for more subject. However, we faced a surprising amount of difficulties with getting our experiment setup working. Neither of us had previously done anything EEG related, so this was an entirely new field for both of us. We had to learn a lot by trial and error. Because of this, getting started took us quite a lot of time. We also had to stay in the scope of a bachelor's thesis. Because we also did not have much experience with VR in Unity, we wanted to have a relative simple VR environment.

This type of an EEG experiment is also something that extensive research has not previously been done on. The traditional oddball experiment has been widely studied and applied, but combining it with VR and asking the subject move their head is something newer. We hope that even though our study ended up being fairly limited and somewhat inconclusive, it might still provide useful information for someone who wants to carry out a similar experiment in the future.

#### *6.2.2. Other Limitations*

As mentioned in the design chapter, we lacked the proper physical equipment to send triggers for both the head movements and the oddball sound cues. That was the reason for implementing the local server setup, which turned out great in the end. However, it created more possible points of error. We cannot be 100% sure whether all head motion triggers were captured and forwarded accurately. Also, all the forwarding of the data might have caused additional fluctuating delay compared to a setup with two separate ports and wires.

However, minor delay is not a concern, especially if it stays somewhat constant. Even if inaccurate, the head motion trigger marks were more than close enough to see which disturbances were head related. And since the focus of this paper was not the brain response itself but the data processing leading to it, perfectly precise timing was not necessary anyway.

Also, as was found out in the data processing chapter, it looked like the head band of the HMD might have been rubbing against the Pz electrode. This probably decreased the quality of our data, especially since the P300 is strongest in the parietal lobe region.

## 6.3. Future Work

There is plenty of room for trying different types of VR applications and experiments combined with EEG. One example being an experiment where the VR application would make sure that the subject moves their head more frequently and for longer periods of time. Designing a good one is a tough task but it would make sure that the brain response portions of the data get properly contaminated with artifacts.

Another topic of research that could utilize EEG would be measuring the eye activity while using VR. What do people do with their eyes and muscles when rotating their head while wearing an HMD? From our experience, people also tend to clench their jaw and tense their face and neck muscles while wearing a relatively heavy HMD, which most likely has an effect on the collected EEG data as well.

Since it looks like the P300 somewhat correlates with the head turns, at least in our experiment, it could be interesting to run this same experiment with either trying to somehow make the P300 and head movements not correlated, or by finding something other than a P300 to look for. It could also be interesting to just run this same experiment with more subjects. Different kinds of data processing pipelines could also be applied.

Even though this study ended up being fairly limited, we hope that this paper might be useful for someone who wants to carry out similar experiments in the future. The possibilities in this area of research seem endless.

# 7. SUMMARY

In the end, it turned out that our experiment design was not entirely successful. But regardless, we still demonstrated a way to remove head movement related components without also removing brain activity from the data.

When it comes to researching ERPs in this manner, careful considerations must be taken with the design of the experiment. If our conclusions are correct, divided attention seems to be able to ruin the P300 component relatively easily.

We could at least determine that ICA is capable of distinguishing head movement related components. But we also found out that the significance of those components is quite low. The results also suggest that a large portion of the head movement artifacts might be included in the eye related components found by ICA, so specifically looking for head movement components might not be necessary very often.

After all, it appears that head movement might not be a huge concern in experiments where both VR and EEG are utilized. However, more research needs to be done on the topic before drawing further conclusions. We hope that this study can inspire new ideas for future research, and might be of use for someone who wants to carry out similar experiments in the future.

# 8. CONTRIBUTIONS

This project was carried out by Lari Kuistio and Joona Meriläinen. Most of the work was done in direct cooperation without specifically assigned tasks. However, some of the contributions are listed below since it was a requirement for this thesis. Both members worked on all parts of the project at least a little bit. However, Joona did more work on the VR environment, and Lari had a bigger role in managing the project.

## 8.1. Contributions of Lari Kuistio

- Project management
- Communication with the supervisor
- Experiment design
- Supervising the experiment session
- Data processing in MatLab
- Writing this paper

## 8.2. Contributions of Joona Meriläinen

- Experiment design
- Development of the VR environment
- Supervising the experiment session
- Data processing in MatLab
- Writing this paper

# 9. REFERENCES

[1] Luck S.J. (2005) An Introduction to the Event-Related Potential Technique. MIT press.

[2] Mumtaz W., Rasheed S. & Irfan A. (2021) Review of challenges associated with the EEG artifact removal methods. Biomedical Signal Processing and Control 68. DOI: https://doi.org/10.1016/j.bspc.2021.102741.

[3] Luck S.J. (2014) An Introduction to the Event-Related Potential Technique, second edition. MIT press.

[4] Bohil C., Alicea B. & Biocca F. (2011) Virtual reality in neuroscience research and therapy. Nat Rev Neurosci. 12(12), pp. 752–762. Doi: 10.1038/nrn3122. PMID: 22048061.

[5] Tromp J., Peeters D., Meyer A.S. & Hagoort P. (2017) The combined use of virtual reality and eeg to study language processing in naturalistic environments .

[6] Hyvärinen A. Karhunen J. O.E. (2001) Independent Component Analysis. John Wiley & Sons Inc.

[7] Iriarte J., Urrestarazu e., Valencia M., Alegre M., Malanda A., Viteri C. & Artieda J. (2003) Independent Component Analysis as a Tool to Eliminate Artifacts in EEG: A Quantitative Study. Journal of clinical neurophysiology: official publication of the American Electroencephalographic Society 20(4). DOI: 10.1097/00004691-200307000-00004.

[8] Vetterli M., Kovacevic J. & Goyal V.K. (2014) Foundations of Signal Processing. Cambridge University Press.

[9] Krokos E. & Varshney A. (2018) Quantifying VR cybersickness using EEG https://doi.org/10.1007/s10055-021-00517-2.

[10] Grassini S., Laumann K., Thorp S. & Topranin V.d.M. (2021) Using electrophysical measures to evaluate the sense of presence in immerse virtual environments: An event-related potential study DOI: https://doi.org/10.1002/brb3.2269.

[11] Baumgartner T., Valko L., Esslen M. & Jäncke L. (2006) Neural Correlate of Spatial Presence in an Arousing and Noninteractive Virtual Reality: An EEG and Psychophysiology Study. CyberPsychology & Behavior 9(1), pp. 30–45. Https://doi.org/10.1089/cpb.2006.9.30.

[12] Bischof W.F. & Boulanger P. (2004) Spatial Navigation in Virtual Reality Environments: An EEG Analysis. CyberPsychology & Behavior 6(5), pp. 487–495. Https://doi.org/10.1089/109493103769710514.

[13] Gratton G., Coles M.G. & Donchin E. (1983) A NEW METHOD FOR OFF-LINE REMOVAL OF OCULAR ARTIFACT. Electroencephalography and clinical Neurophysiology 55, pp. 468–484. DOI: https://doi.org/10.1016/j.tins.2008.11.001.

[14] Ochoa C.J. & Polich J. (2000) P300 and blink instructions. Clinical Neurophysiology 111.

[15] O' Regan S., Faul S. & Marnane W. (2010) Automatic detection of eeg artefacts arising from head movements , pp. 6353–6356.

[16] Tremmel C., Herff C. & Krusienski D.J. (2019) Eeg movement artifact suppression in interactive virtual reality , pp. 4576–4579.

[17] Chen X., Xu X., Liu A., Lee S., Chen X., Zhang X., McKeown M.J. & Wang Z.J. (2019) Removal of Muscle Artifacts From the EEG:A Review and Recommendations. IEEE SENSORS JOURNAL 19(14).

[18] Stone J.V. (2002) Independent component analysis: an introduction. TRENDS in Cognitive Sciences 6(2).

[19] Cao X.R. & Liu R.W. (1996) General approach to blind source separation. IEEE Transactions on Signal Processing 44, pp. 562–571.

[20] Makeig S. & Onton J. (2012) ERP features and EEG dynamics: An ICA perspective DOI: 10.1093/oxfordhb/9780195374148.013.0035.

[21] Hoffmann S. & Falkenstein M. (2008) The Correction of Eye Blink Artefacts in the EEG: A Comparison of Two Prominent Methods Doi:10.1371/journal.pone.0003004.

[22] Urigüen J.A. & Garcia-Zapirain B. (2015) EEG artifact removal—state-of-the-art and guidelines. Journal of Neural Engineering 12, p. 031001. URL: `https://doi.org/10.1088/1741-2560/12/3/031001`.

[23] Xue Z., Li J., Li S. & Wan B. (2006) Using ica to remove eye blink and power line artifacts in eeg. In: First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06), vol. 3, vol. 3, pp. 107–110.

[24] Picton T.W. (1992) The P300 Wave of the Human Event-Related Potential. Journal of Clinical Neurophysiology 9(4), pp. 456–479. URL: `https://journals.lww.com/clinicalneurophys/Abstract/1992/10000/The_P300_Wave_of_the_Human_Event_Related_Potential.2.aspx`.

[25] Polich J. (2007) Updating p300: An integrative theory of p3a and p3b. Clinical Neurophysiology 118, pp. 2128–2148. URL: `https://www.sciencedirect.com/science/article/pii/S1388245707001897`.

[26] Peirce J., Gray J. & Simpson S.e.a. (2019) Psychopy2: Experiments in behavior made easy. Behavior Research Methods 51, pp. 195–203. URL: `https://sccn.ucsd.edu/eeglab/download/eeglab_jnm03.pdf`.

[27] Delorme A. & Makeig S. (2004) Eeglab: an open-source toolbox for analysis of single-trial eeg dynamics. Journal of Neuroscience Methods 134, pp. 9–21. URL: `https://sccn.ucsd.edu/eeglab/download/eeglab_jnm03.pdf`.

[28] Lopez-Calderon J. & Luck S.J. (2014) Erplab: An open-source toolbox for the analysis of event-related potentials. Frontiers in human neuroscience 8:231. URL: `https://www.frontiersin.org/articles/10.3389/fnhum.2014.00213/full`.

[29] Delorme A., Palmer J., Onton J., Oostenveld R. & Makeig S. (2012) Independent eeg sources are dipolar. PLOS ONE 7, pp. 1–14. URL: `https://doi.org/10.1371/journal.pone.0030135`.

[30] Frederik Nagel (2023). Point biserial correlation. MATLAB Central File Exchange, howpublished = `https://www.mathworks.com/matlabcentral/fileexchange/11222-point-biserial-correlation`, note = Accessed: 2023-02-02.

# 10. APPENDIX

## Appendix A: Code for the trigger server

```
import socket
import serial
import time


def send_trigger(d):
    """
    Writes a hexadecimal to the port and waits a certain time
    to allow the write to finish. Then writes a zero to the port to reset it.
    d: integer for the trigger mark (e.g. d=4 adds an 'S4' to the EEG data)
    """

    if d == 0:
        port.write([0x00])
        print("{} sent".format(d))
    elif d == 1:
        port.write([0x01])
        print("{} sent".format(d))
    elif d == 2:
        port.write([0x02])
        print("{} sent".format(d))
    elif d == 3:
        port.write([0x03])
        print("{} sent".format(d))
    elif d == 4:
        port.write([0x04])
        print("{} sent".format(d))
    elif d == 5:
        port.write([0x05])
        print("{} sent".format(d))
    elif d == 6:
        port.write([0x06])
        print("{} sent".format(d))

    time.sleep(.005)
    port.write([0x00])
    time.sleep(.005)


# Set up socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(("localhost", 9988)) # IP = localhost / 127.0.0.1;
                            # Socket = 9988
                            # (make sure that clients use these for sending!)


# Set up com port
port = serial.Serial("COM4")

print("Server started.")

while True:
    message, address = s.recvfrom(16)
    send_trigger(int(message.decode("utf-8")))
```

## Appendix B: Rotation detection script used with Unity

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using System;
using System.IO;
using System.IO.Ports;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.Text;

public class RotationDetectionScript : MonoBehaviour
{
    // Queues for orientation samples collected
    Queue<float> orientationX = new Queue<float>();
    Queue<float> orientationY = new Queue<float>();
    Queue<float> orientationZ = new Queue<float>();

    // Booleans to set what is done during a session
    private bool sendTriggers = false;
    public bool sendServer = true;

    // Interval for orientation data collection
    private float interval = 0.05f;

    // Threshold above which the averaged values should get to for a head motion to
        be detected
    private float threshold = 0.6f;

    // Threshold below which the averaged values should get for the end of a head
        motion to be detected
    private float lowThreshold = 0.3f;

    // Average of the substractions of the queue values
    private float xAverage = 0;
    private float yAverage = 0;
    private float zAverage = 0;

    // Boolean which is true during a head motion
    private bool turning = false;

    // Serial port for sending the triggers if done directtly
    SerialPort serialPort = new SerialPort("COM4", 115200, Parity.None, 8, StopBits.
        One);

    // A separate thread is needed for sending the triggers directly through a
        serialport
    Thread serialT;

    // Socket for sending the triggers if done via a server
    Socket sock;
    IPEndPoint endPoint;

    void Start() {
        // If triggers are sent directly via a serial port
        if(sendTriggers == true) {
            // CLose the port if it was left open for some reason
            if (serialPort.IsOpen)
                serialPort.Close ();

            // Open the port
            serialPort.Open();

            // Timeout for writing to the port incase it fails or takes too long
            serialPort.WriteTimeout = 100;

            // Reset the port by writing zero to it
            byte[] data = new byte[] { 0 };
            serialPort.Write(data, 0, data.Length);
        }

        // Initialize queues with zeros to get the right lengths for them
        for (int i = 0; i < 20; i++) {
            orientationX.Enqueue(0);
            orientationY.Enqueue(0);
            orientationZ.Enqueue(0);
        }
    }

    void Update()
    {
        /* Update is called every frame but we have a coroutine to create a delay
            which is the interval for the orientation samples */
        /* E.g. If the interval is 0.5s, we collect the orientation data every 0.5s
            which leads to this function being called every 0.5s */

        // Deque the oldest orientation samples and sample new ones
        dequeueAll();
        enqueueAll();

        // Calculate the averages for the orientation deltas
        xAverage = average(orientationX);
        yAverage = average(orientationY);
        zAverage = average(orientationZ);

        if (turning == false) {
            if (xAverage > threshold) {
                print("turn start x " + xAverage);
                turning = true;

                if (sendServer == true) {
                    sendToServer(5);
                }


                if(sendTriggers == true) {
                    //print("sent");
                    serialT = new Thread(() => writeSerial(serialPort, 5));
                    serialT.Start();
                }
            } else if (yAverage > threshold) {
                print("turn start y " + yAverage);
                turning = true;

                if (sendServer == true) {
                    sendToServer(5);
                }

                if(sendTriggers == true) {
                    //print("sent");
                    serialT = new Thread(() => writeSerial(serialPort, 5));
                    serialT.Start();
                }
            } else if (zAverage > threshold) {
                print("turn start z " + zAverage);
                turning = true;

                if (sendServer == true) {
                    sendToServer(5);
                }

                if(sendTriggers == true) {
                    //print("sent");
                    serialT = new Thread(() => writeSerial(serialPort, 5));
                    serialT.Start();
                }
            }
        } else {
            if (xAverage < lowThreshold && yAverage < lowThreshold && zAverage <
                lowThreshold) {
```

```
                print("turn end");
                turning = false;

                if (sendServer == true) {
                    sendToServer(6);
                }

                if(sendTriggers == true) {
                    //print("sent");
                    serialT = new Thread(() => writeSerial(serialPort, 6));
                    serialT.Start();
                }
            }
        }

        // Interval
        StartCoroutine (waitInterval());
    }

    void sendToServer(int i) {
        // Send the trigger information to the server which sends the data to the
            triggerbox

        // Set up the UDP socket
        sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.
            Udp);

        // IPddress of the server
        IPAddress serverAddr = IPAddress.Parse("127.0.0.1");

        // Socket number
        endPoint = new IPEndPoint(serverAddr, 9988);

        string text = "";

        // Set the data accordingly
        if (i == 5) {
            text = "5";
        } else if (i == 6){
            text = "6";
        } else {
            text = "0";
        }

        // Send data via UDP
        byte[] send_buffer = Encoding.ASCII.GetBytes(text);
        sock.SendTo(send_buffer, endPoint);
    }

    void writeSerial(SerialPort s, int i) {
        // Send the trigger depending on the 'i' that is passed to this function

        if (i == 0) {
            byte[] data = new byte[] { 0 };
            s.Write(data, 0, data.Length);
            Thread.Sleep(100);    // Allow some time for the writing
            byte[] data1 = new byte[] { 0 }; // Reset port after use
            s.Write(data1, 0, data1.Length);
        } else if (i == 5) {
            byte[] data = new byte[] { 5 };
            s.Write(data, 0, data.Length);
            Thread.Sleep(100);
            byte[] data1 = new byte[] { 0 };
            s.Write(data1, 0, data1.Length);
        } else if (i == 6) {
            byte[] data = new byte[] { 6 };
            s.Write(data, 0, data.Length);
            Thread.Sleep(100);
            byte[] data1 = new byte[] { 0 };
            s.Write(data1, 0, data1.Length);
        }
    }

    void dequeueAll() {
        // Dequeue the oldest values from each of the lists
        orientationX.Dequeue();
        orientationY.Dequeue();
        orientationZ.Dequeue();
    }

    void enqueueAll() {
        // Sample orietation values and enqueue them into the lists
        // Orientation values are between 0-360, but we want them to be between -180
            and 180

        // Sample x-axis orientation
        if (gameObject.transform.rotation.eulerAngles.x <= 180) {
            orientationX.Enqueue(gameObject.transform.rotation.eulerAngles.x);
        } else {
            orientationX.Enqueue(gameObject.transform.rotation.eulerAngles.x - 360);
        }

        // Sample y-axis orientation
        if (gameObject.transform.rotation.eulerAngles.y <= 180) {
            orientationY.Enqueue(gameObject.transform.rotation.eulerAngles.y);
        } else {
            orientationY.Enqueue(gameObject.transform.rotation.eulerAngles.y - 360);
        }

        // Sample z-axis orientation
        if (gameObject.transform.rotation.eulerAngles.z <= 180) {
            orientationZ.Enqueue(gameObject.transform.rotation.eulerAngles.z);
        } else {
            orientationZ.Enqueue(gameObject.transform.rotation.eulerAngles.z - 360);
        }
    }

    float average(Queue<float> q) {
        // Calculate the average of the deltas between subsequent orientation samples
        // The orientation sample list is passed to this function
        // Returns the average of the deltas

        // List for the deltas (E.g. a list of 20 samples has 19 deltas)
        float[] subtractions = new float[19];

        // Substract the subsequent value in the list which are the deltas
        for (int i = 0; i < 19; i++) {
            subtractions[i] = Math.Abs(q.ElementAt(i+1) - q.ElementAt(i));
        }

        float sum = 0;

        // Sum the deltas for the average calculation
        for (int i = 0; i < 19; i++) {
            sum += subtractions[i];
        }

        // Return the average
        return sum/19;
    }

    IEnumerator waitInterval() {
        // Delay for the update function in order to have an interval for the
            orientation sampling
        yield return new WaitForSeconds (interval);
    }

}
```

## Appendix C: Script used with Unity to switch the TV's

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO.Ports;

public class TVSwitchScript : MonoBehaviour
{
    public Material onMaterial;
    public Material offMaterial;
    public bool onOff;
    public GameObject L;
    public int lowTimerValue = 0;
    public int highTimerValue = 1;
    public int cycleCount = 5;
    private int cycles = 0;

    public GameObject x;

    void Start() {
        cycles = 0;
        StartCoroutine (startWaiter());
    }

    void Change () {
        if (onOff == true) {
            onOff = false;
            GetComponent<MeshRenderer >().material = offMaterial;
            L.GetComponent<MeshRenderer >().material = onMaterial;
        } else {
            onOff = true;
            GetComponent<MeshRenderer >().material = onMaterial;
            L.GetComponent<MeshRenderer >().material = offMaterial;
        }
        cycles++;
        if (cycles < cycleCount) {
            StartCoroutine (waiter());
        } else {
            StartCoroutine (startWaiter());
        }

    }

    IEnumerator waiter() {
        int waitTime = Random.Range (lowTimerValue, highTimerValue);
        yield return new WaitForSeconds (waitTime);
        print ("SWITCH " + "(" + waitTime + ")");
        Change();
    }

    IEnumerator startWaiter() {
        yield return new WaitForSeconds (5);
        if (cycles == 0) {
            print ("START");
            Change();
        } else {
            print ("END");
        }
    }
}
```

Appendix D: Matlab correlator script used to compute the correlations between ICA components and head turn markers. The pointbiserial function is written by Frederik Nagel [30].

```
% use with continuous data once ICA weights have been computed
corr_mat = [];

% if using EEG channel
mastoid_chan = 21;

% if using head turn triggers
turn_channel = zeros(length(EEG.times),1); % empty vector which we will fill with 1s
    and 0s to signify ongoing turns
turn_start = 0; % has the person started turning their head?
turn_end = 0; % have they stopped turning their head?

for i = 1:length(EEG.event)
    % wait for turn onset and note the latency
    if strcmp(EEG.event(i).type, 'S  5')
        turn_start = 1;
        t_turn_start = int64(EEG.event(i).latency);
    end
    % wait for turn offset and note the latency
    if strcmp(EEG.event(i).type, 'S  6')
        turn_end = 1;
        t_turn_end = int64(EEG.event(i).latency);
    end
    % once onset and offset are complete, fill from turn onset until turn
    % offset with 1 to indicate presence of turn
    if turn_start == 1 && turn_end == 1
        turn_channel(t_turn_start:t_turn_end) = 1;
        turn_start = 0;
        turn_end = 0;
    end
end

% get correlation coefficients between ICA components and turn signals
for i = 1:size(EEG.icaact,1)
    [r,h,p,ci] = pointbiserial(turn_channel,abs(EEG.icaact(i,:)));
    corr_mat(i,1) = abs(r);
    corr_mat(i,2) = p;
    if p < 1*10^-20
        corr_mat(i,3) = 1;
    else corr_mat(i,3) = 0;
    end
    corr_mat(i,4) = abs(ci(1));
    corr_mat(i,5) = abs(ci(2));
end

function [r,h,p,ci] = pointbiserial(d,x,alpha,tail)
%POINTBISERIAL Calculate Point biserial correlation
%       R = POINTBISERIAL(X,Y) calculates the correlation with
%   D logical (boolean) variable, values 0 or 1. If values are numeric, all
%      values ~=0 are set to 0 and
%   X as continuos variable.
%
%   [R,H,P,CI] = POINTBISERIAL(D,X)
%   H = 1 means that you can reject the null hypothesis, the means are
%   equal at the 5% significance level. P holds the p-value associated
%   with the t-statistic, CI is a 95% confidence interval for the true
%   difference in means.
%   A t-test is computed to check for the difference of both groups. Note
%   that a normal distribution is desired for the computation of the
%   t-test. For a non-parametric test use the 4-parameter call below.
%
%   [...] = POINTBISERIAL(D,X,ALPHA) can be used to set a different value for
%   alpha. Default is 5%.
%
%   [...] = POINTBISERIAL(D,X,ALPHA,TAIL) performs the test against the alternative
%   hypothesis specified by TAIL: (taken from ttest2)
%         'both'  -- "means are not equal" (two-tailed test)
%         'right' -- "mean of X with D == 1 is greater than mean of X with D == 0" (
     right-tailed test)
%         'left'  -- "mean of X with D == 1 is less than mean of X with D == 0" (left-
     tailed test)
%
%    [R,H,P,STATS] = POINTBISERIAL(D,X,ALPHA,'np') performs a non-parametric Wilcoxon
     rank sum
%    test (2-tailed). STATS contains test result struct (see ranksum).
%
% Example:
%    parm = rand(100,1);
%    gender = rand(100,1);
%    gender(gender<=.5)=0;
%         [r,h] = pointbiserial(gender,parm,.05);
% Uses:
%         Matlab Statistics Toolbox
%
% Frederik Nagel
% Institute of Music Physiology and Musicians' Medicine
% Hanover University of Music and Drama
% Hannover
% Germany
%
% e-mail: frederik.nagel@hmt-hannover.de
% homepage: http://www.immm.hmt-hannover.de
%
% May 29, 2006.
%
% See also CORR, TTEST2, RANKSUM

error(nargchk(2, 4, nargin))
if(nargin==2)
     alpha=.05;
     tail = [];
elseif(nargin==3)
     tail = [];
end

% Convert numeric values to logicals
d = logical(d);

% Length
n = length(d);

% Lengths of groups 0 and 1
n1 = sum(d);
n0 = sum(~d);
if(n0==0)
     error('There are no data with x=0!');
elseif(n1==0)
     error('There are no data with x=1!');
elseif (n0+n1 ~= n || sum(isnan(d))>0 || sum(isnan(x))>0)
     error('Data may not contain NANs');
end

% Mean of groups 0 and 1
x1 = mean(x(d));
x0 = mean(x(~d));

% SD of y
sx   = std(x);

%Correlation coefficient
r = (x1-x0)/sx*sqrt (n0*n1/n^2);

if(isempty(tail))
     [h,p,ci] = ttest2(x(d),x(~d));
elseif(strcmp(tail,'np'))
     [p,h,ci] = ranksum(x(d),x(~d),alpha);
else
```

```
    [h,p,ci] = ttest2(x(d),x(~d),alpha,tail);
end
```