**Faculty of Social Sciences**
Artificial Intelligence

Academic year 2014-2015
Date 08-07-2015

# Using Fixation Related Potentials to Explore the Difference Between Cancellations and Re-cancellations During Visual Search

Danny Merkx, s0813400
Supervisors: Dr. Ervin Poljac & Marjolein van der Waal

Radboud University

## Abstract

*Purpose* - Deficits in spatial working memory (SWM) are thought to aggravate the symptoms of neglect. This study investigates brain activity during a visual search task, a task which depends on SWM, in order to better understand the role of SWM during such tasks.

*Design* - Using EEG, four healthy subjects were recorded while performing a visual search task in which target stimuli had to be found in between distractor stimuli. An eye-tracker was used to determine which stimulus the subject was fixating on.

*Findings* - Cancellations could be distinguished from re-cancellations well above chance for one out of four participants (59%). Two others performed slightly above chance (53%) and for one participant performance was at chance level.

*Possible limitations* - It seems that a distinction between cancellations and re-cancellations based on EEG data can be made but improvements to the task paradigm are needed to gather more and higher quality FRPs in order to get a better overall performance. Improvements to the current paradigm were suggested and implemented.

*Originality/value* - Very little is known about the cognitive deficits underlying spatial neglect syndrome. This study attempts to increase our understanding by looking at the brain activity of healthy subjects during a task that neglect patients typically do not perform well on. This study can provide novel insights into what happens during the processing of targets in healthy subjects. As such it can provide data for future studies on neglect patients to compare to.

# Contents


**1 Introduction** **3**
- 1.1 Research Objectives . . . 3
- 1.2 Relevance . . . 4

**2 Background** **4**
- 2.1 Electroencephalography . . . 4
- 2.2 Event Related Potentials . . . 5
- 2.3 Fixation Related Potentials . . . 5
- 2.4 Electrooculography . . . 6
- 2.5 Spatial Working Memory . . . 6
- 2.6 Noise and Artefacts . . . 6

**3 Methods** **8**
- 3.1 Participants . . . 8
- 3.2 Equipment . . . 8
- 3.3 Stimuli . . . 8
- 3.4 Task Design and Procedure . . . 9

**4 Analysis** **10**
- 4.1 Fixation Processing . . . 10
- 4.2 EEG Pre-processing . . . 10
- 4.3 Classification . . . 10

**5 Results** **11**

**6 Discussion** **12**
- 6.1 Possible Limitations . . . 12
- 6.2 Suggested Improvements . . . 13

**7 Conclusion** **14**

**8 Acknowledgements** **14**

**9 Literature** **15**

**10 Appendices** **18**
- 10.1 Appendix A . . . 18
- 10.2 Appendix B . . . 24
- 10.3 Appendix C . . . 26
- 10.4 Appendix D . . . 31

# 1 Introduction

Spatial neglect syndrome (from here on neglect) is a neurological disorder resulting from Acquired Brain Injury (ABI) such as a stroke. Studies report incident rates of up to 55% in right and 20% in left hemisphere stroke [1-3]. Neglect is a multi-modal disorder characterised by "the failure to report, respond, or orient to meaningful or novel stimuli"[4]. Though not explicit in this definition, an important characteristic is that it is not a sensory or motor deficit [5]. Patients fail to react to stimuli as opposed to not perceiving them (sensory deficit e.g. blindness) or being physically inable to react (motor deficit e.g. paralyses). Patients display symptoms like eating only from one side of a plate, reading incomplete sentences and an impaired ability to avoid obstacles [7][8]. The inability to react to one side of one's world is a major cause of disability and the severity of cognitive deficits, even more so than stroke severity itself, is predictive of performance in daily life activities [2][6].

Several types of therapies and rehabilitation programs have been developed and tested with mixed results, or lack assessment of long term effectiveness (see [5] and [7] for reviews). Many of these programs focus mainly on impaired spatial attention (i.e. failure to explore the neglected hemisphere), for instance in Prism Adaptation, in which special prismatic goggles adjust the field of vision towards the neglected hemisphere [9][10], or by using special glasses that produce auditory feedback to draw attention towards neglected stimuli [11]. Recent research shows that neglect is a heterogeneous disorder and that only addressing one of its components might not be enough to a restore patients cognitive abilities. Though neglect is traditionally characterised by a deficit in spatial attention, research pointed to other components of the disorder such as impaired temporal allocation of attention [12] and impaired spatial working memory (SWM)[13][14]. It has been argued that addressing SWM in therapies might be vital to improving the effectiveness of treatments [15].

As of yet, very little is known about the role of SWM in neglect and even less about the underlying neural correlates of (impaired) SWM. In a recent study, Wansard et al. have separately assessed SWM and performance in visual search in neglect patients, and found that impaired SWM is related to re-visiting (re-cancelling) of cancelled targets [14]. In such cancellation tasks, the participant has to find all targets usually in between non-targets, and when a target is found this is called a cancellation. These tasks require SWM as the participant has to remember the location of the targets they have already found. However, Wansard et al. have not investigated the underlying brain pathology and as such claim that it is not clear if the results are due to impaired SWM or due to compromised executive functions like planning (i.e. the information is retained in memory but not acted upon).

A review of neglect studies that utilise EEG also concludes that it is as of yet unknown at what stage the normal processing breaks down in patients, and answering this question is an important step in improving rehabilitation and diagnoses [16]. Other research suggest that the mechanisms of spatial attention and SWM are related, and that spatial attention has a rehearsal like function that keeps information active in working memory [17]. This could mean the impaired SWM is a result of impaired spatial attention or that spatial attention impairments exacerbate dysfunction of the SWM. EEG and other brain imaging techniques have not yet been used to compare the brain activity of healthy subjects and patients during a task that requires SWM. Such research might increase our understanding of SWM impairments

## 1.1 Research Objectives

In the current study fixation related potentials (FRP) from an electroencephalogram (EEG) recorded during a visual search task are used to investigate the neural correlates of SWM. FRPs are a combination of the traditional event related potentials (ERP) and eye-tracking. Whereas in ERPs the brain potentials are time locked to stimulus presentation, FRPs are time locked to the subjects eye fixation on a stimulus.

In a visual search paradigm a cancellation occurs when a subject has found a target stimulus. When multiple targets are present in a single trial and the cancelled target is not marked in any way, the subject is thus required to remember the location of the cancelled target (and as such it involves SWM)[18]. When the subject revisits a target this is called a re-cancellation. This can be paired with a motor reaction such as clicking on the target, or indication whether a target is new through a button box. In this research there will be no motor reaction to stimuli, but cancellations will refer to the first fixation on each target and re-cancellations to any subsequent fixation on a target.

The main focus of this study is comparing brain responses in cancellations with those in re-

cancellations. A comparison between target and non-target FRPs will also be made to confirm if the data shows the expected differences in P300 magnitude that are reported by other studies [19][20].

Very little is known about FRPs in combination with SWM in both neglect patients and healthy subjects and data is absent for both groups. This study will focus on gathering data from healthy subjects only.

The goal of this research is:

> To distinguish between cancellations (first fixations) and re-cancellations (subsequent fixations) during a visual search task using a combination of EEG and eye-tracking.

### 1.2 Relevance

In an aging society, strokes and the ensuing disabilities like neglect will incur great societal costs [21][22]. Finding proper ways to rehabilitate stroke survivors and increase their independence from care givers is important in reducing strain on the medical system.

Current neglect therapies still lack evidence for their effectiveness and research indicates that incorporating training for other neglect related deficits like SWM in rehabilitation programs might improve their chances of success [15]. By comparing cancellation and re-cancellation FRPs we can learn how the brain responses to these events differ. This gives us more information on the workings of SWM , which is important for creating rehabilitation programs that target this cognitive function.

Furthermore, knowledge of FRPs can be applied outside of the field of medicine. Systems like brain computer interfaces (BCIs) are controlled by the users brain signals and the system classifies these to predict the user's intention. In order to do this however, we need to know how to elicit brain responses from the user that reflect their intention. The ability to distinguish between cancellations and re-cancellations can also be used to support visual search tasks, for instance in speeding up screening processes by notifying the user when they keep revisiting old targets.

## 2 Background

### 2.1 Electroencephalography

Electroencephalography or EEG is a method to record the brain's electrical activity using electrodes placed along the scalp [23]. It is non-invasive meaning the electrodes are placed on the scalp, typically in a cap, as opposed to for example electrocorticography (ECoG) where electrodes are surgically implanted in the brain. An EEG represents changes in the brains electrical activity as measured in Volt. Figure 1 shows the equipment used for the EEG and other data recording in the current study.

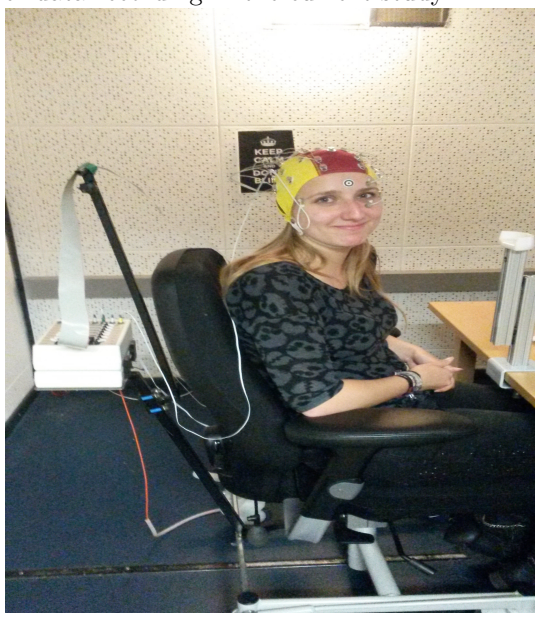


**Figure 1: Participant fitted with EEG cap, EOG electrodes around the eyes and target sticker on the forehead for the eye-tracker.**

Neurons in the brain maintain a so called resting potential, where the inner surface of the cell's membrane has an electrical charge of about -70 mV relative to the outer surface of the membrane. In the absence of any stimulation this charge remains steady. When a neuron is electrically stimulated this resting potential is disturbed and the polarity of the intra- and extracellular charge changes. Now the inside of the membrane has a positive charge relative to the outside of the membrane for about a millisecond. These so called action potentials lead the neuron to release neurotransmitters which in turn stimulates it's neighbours and so passing on the action potential. This means the brains electrical activity is propagated by the neurons rather than flowing like a current. After an action potential the neuron returns

to the resting potential but the short nature of this cycle means a neuron can fire many times in a single second.

These action potentials are small with the total change in of voltage little of 100 mV. Thousand to millions of neurons need to be active for the activity to be measured by the scalp electrodes used in EEG. This also limits EEG to recording cortical activity as it cannot reliably measure activity in deeper brain structures.

Figure 2 shows what an EEG is recording. It is one second of EEG data for 32 channels that was collected for this research. Visual inspection like this can be useful during the experiment for instance to detect bad channels.

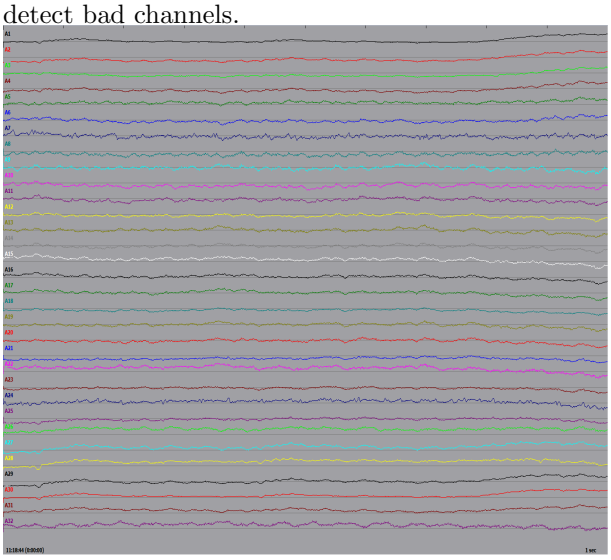


**Figure 2: EEG; each line represents the changes in electrical activity measured by the corresponding electrode.**

## 2.2 Event Related Potentials

When studying the functioning of the brain using EEG we make use of event related potentials (ERP)[23][24]. An ERP is a brain response as a direct result of some sensory stimulation. In experiments this event is a carefully chosen stimulus and EEG is used to record the participants brain response to the stimulus presentation. The ERP is time-locked to the onset of the stimulus and this onset is used to select the appropriate data from the EEG.

It takes a large amount of measurements per condition to determine the waveform of an ERP. An EEG contains random brain activity and noise besides that which is of interest. However these random components are unique while the ERP waveform occurs in each sample, thus with a large sample the random components get averaged out while the ERP remains as seen in figure 3.

A well known component of an ERP is the P300 named so because it a positive potential and it occurs roughly 300 ms after a stimulus presentation [25]. An example application of the P300 is in P300 spellers [26]. The user of such a speller is rapidly presented with the letters of the alphabet and the EEG can be used to determine the user's intention based on the stronger P300 component in the ERP following the onset of the users intended letter. So by focussing their attention on the task relevant stimulus, users can modulate their brain signals in order to control a machine.

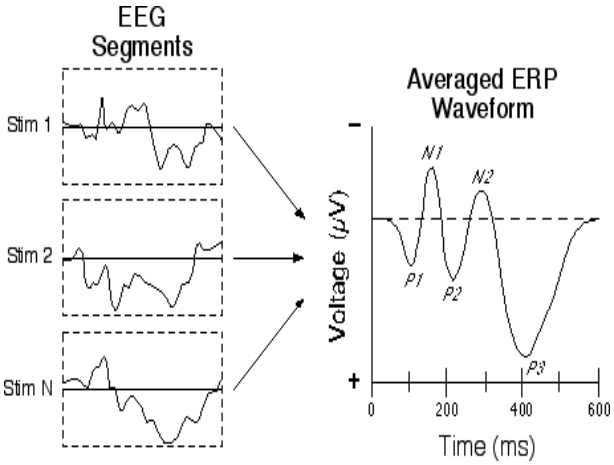



**Figure 3: Averaging a large number of ERPs contaminated with random brain activity and noise reveals the ERP waveform.**[1]

## 2.3 Fixation Related Potentials

As event related potentials are analysed by taking the EEG data following a stimulus stimulus presentation this requires stimuli to be sequentially presented to subjects. This poses a challenge to tasks where multiple stimuli need to be presented at the same time such as in visual search or reading tasks. When the subject is presented with a picture and needs to search all targets in this picture, the stimulus presentation is not a suitable reference point for the ERPs.

This problem can be solved by combining EEG with eye-tracking [24]. By tracking the participants eyes and determining when and where fixations occur it can determined when a participant fixates on

[1] Image source: http://erpinfo.org/what-is-an-erp

a location on screen. The eye-tracker data is used to make a list of fixations with the time of fixation onset and the screen coordinates of these fixations. With this information it can be determined when a participant is looking at a target. The fixation onset rather than the stimulus onset can then be used to extract the appropriate potentials from the data to analyse. These potentials are called fixation related potentials (FRP) due to being the brain's response to eye fixations (see [27] and [28] for earlier research using FRPs in a visual search task). We say that the FRP is time-locked to fixation on a stimulus instead of being time-locked to stimulus presentation as in ERPs.

## 2.4 Electrooculography

Electrooculography (EOG) measures the electric potentials of the eye. Using four electrodes placed around the eyes, eye-movements can be detected. This is possible because the eye acts as an electric dipole [29]. This means the poles of the eye have opposite charges; the cornea has a positive charge while the retina has a negative charge. When the eye moves, for instance when the participant looks up, the cornea's positive charge moves closer to the upper electrode and a positive change in electric potentials is recorded by the electrodes. This is illustrated in the figure below, where it is clearly seen that a positive trend in one electrode is coupled with a negative trend in the opposite electrode.

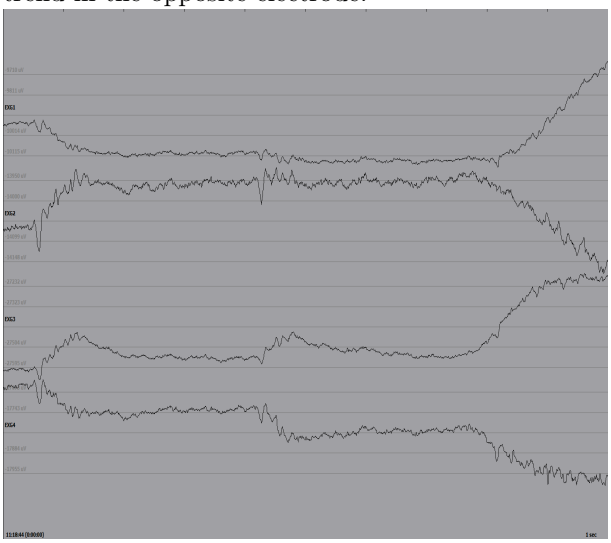


**Figure 4: EOG; the four lines represent electrodes next to the right and left eye and those above and under the left eye respectively.**

When it is necessary for participants to move their eyes during an experiment the movement of the eye's electric potential fields is also picked up by the EEG. In the current research EOG data is gathered to remove eye-movement artefacts from the EEG data. The EOG is used as a so called reference channel; when one channel contains EEG activity contaminated with EOG activity, and the reference contains only EOG activity, the reference can be regressed out of the main channel to leave a clean EEG signal [30][31].

## 2.5 Spatial Working Memory

SWM is a cognitive function that is used in short-term storage of spatial information (i.e. information about an object's location in space) and in the mental manipulation of spatial information, such as when judging whether two sequentially displayed lines are aligned [32].

Patients with neglect often suffer from impaired spatial working memory exacerbating their symptoms and posing difficulties to rehabilitation [14]. SWM plays an important role in visual search, a task we all perform daily as we search for our keys or look for a familiar face in a crowd for instance. The role of SWM in visual search is confirmed in recent research. Striemer et al. had a neglect patient perform a visual search task with and without visible cancellation marking, and found that the patient made more re-cancellations in the condition without markings and was treating already visited targets as new [15]. This was confirmed in another study by Wojciulik et al. who used a setup of targets with memorable and unique features and non-unique targets which differed only in location [33]. They found that neglect patients had considerably more re-cancellations for the non-unique targets but not for the unique ones, suggesting that only SWM is affected while working memory remains intact.

In the current research the goal is to investigate the brain signals associated with SWM. For the experimental task to be approriate to investigate this, it is important that the taskrequires SWM meaning there can be no visual marking of cancellations and the targets can have no unique qualities besides their location.

## 2.6 Noise and Artefacts

One of the challenges facing EEG research is the small signal-to-noise ratio and the wide range of sources that produce noise. EEG is used to measure brain

activity and any electrical activity from other sources is considered noise (also called artefacts). The electrodes used to make an EEG need to be very sensitive in order to pick up the cerebral activity, but this also means it can easily pick up all kinds of electrical activity generated by the participant or equipment. A few common sources of artefacts will be described here, for more thorough reviews see [34][36].

Artefacts can be divided in two categories based on their source; physiological artefacts originate from within the subject while extra-physiological artefacts originate from external sources like equipment or the environment. Physiological noise includes EOG activity as described earlier, but also electromyographic (EMG) activity like clenching of the jaw or head movements. While movements can often be avoided by instructing the participant to remain as still as possible, elecrocardiographic (ECG) generated by the contractions of the heart cannot be avoided. Some EEG activity can even be considered noise, for instance when the subject is distracted by a ringing phone during the experiment. This distraction generates an ERP as sure as any genuine stimulus would, but while it is technically EEG activity, it is unwanted in the context of an experiment.

External sources of noise also require careful consideration as this includes the equipment that is necessary to perform the experiment. A well known source of noise is the so called 50 Hz mains hum. Mains power uses alternate current (AC) at 50 Hz, meaning the current reverses direction 50 times per second. This produces a 50 Hz signal in all equipment connected to mains power which can be picked up by the EEG. Another example is the sudden discharge of energy that is built up in the conductive gel between the scalp and the electrode. This is called an electrode pop and generates a strong spike followed by a slower descent.

Without taking measures to prevent or remove noise from the signal, large section of data might be become unusable for analysis. If noise resembling real EEG activity goes unnoticed, the research might even lead to false conclusions, an even greater danger considering EEG research is often involved in medical diagnoses or designing rehabilitation programs for patients.

Not surprisingly many countermeasures have been developed over the years. While some are as simple as instructing participants not make unnecessary movements, others involve specialised algorithms to automatically detect and remove certain types of artefacts. In a review of noise reduction techniques, Repovš identifies four categories of which examples used in this research will be given; elimination of noise sources, signal averaging, removal of noise and rejection of noisy data [36]

As said the easiest way to counter noise is to simply prevent it. It is commonplace to run experiments in an electrically shielded vault, to prevent ambient electrical fields from being picked up by the EEG but also to prevent distractions like unwanted sounds and scenery. The EEG apparatus used in the current study runs on a battery which runs on direct current (DC) as opposed to AC to prevent mains hum. It is important to design the experimental task in such a way as to minimise necessary movement, and if unavoidable to make the movements easy to remove from the data. In the current task the only movement occurs at the end of a trial when the participant clicks the mouse to proceed to the next trial so it is known that the last seconds of a trial are unusable for the analysis.

Still, noise will unavoidably contaminate the data so it will have to be processed to remove noise before it can be used in the analysis. As shown before, regression techniques can be used to remove EOG artefacts from the data. By applying a low pass filter all frequencies above a certain cut-off point are removed from the data. This is a useful technique as for instance the frequency of muscle contractions lies well above the frequency of brain activity and mechanical noise like the mains hum is restricted to exactly 50 Hz. High pass filters are also applied to remove aliases of signals higher than the sampling frequency. For example, when sampling a 4000 Hz signal at 2000 Hz it will be reconstructed in the data as an artificial 2000 Hz signal. To illustrate this somewhat complex concept, in figure 5 you can see how two waves fit the same sample set. When the high frequency signal is sampled at an inadequate sampling rate an alias with half its frequency can be fitted onto the sample. So the EEG is sampled at rate well above the frequencies of the brain to avoid artificial signals as a result of aliasing in the range of interest.

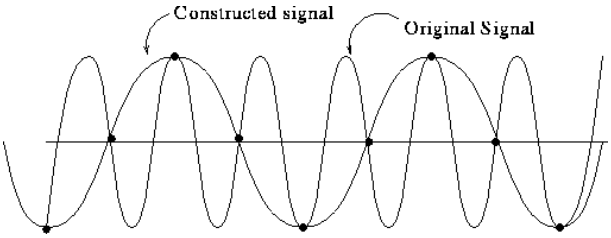



**Figure 5: Aliasing, the dots represent the samples, aliasing occurs when multiple signals fit the same sample set.**[2]

There are still artefacts within the bandwidth of brain activity however, and sometimes rejection of noisy samples is unavoidable. Lastly the remaining random noise will be averaged out of the FRPs as long as the noise is random and enough samples are available.

# 3 Methods

## 3.1 Participants

Four volunteers (2 males and 2 females, aged 22–28, mean age 25) participated in the experiment. The procedure was explained to the participants but they were kept naive as to the objectives of the experiment. This is important as knowledge of the research objectives might influence their eye-gaze behaviour. Participants signed an informed consent form before the start of the procedure.

## 3.2 Equipment

Stimuli were presented on a 34 x 27 cm flat-screen Phillips 170B monitor, with a resolution of 800 x 600 pixels and a refresh rate of 60 Hz.

Eye-movement was measured at 500 Hz using an EyeLink 1000 eye-tracker[3]. The EyeLink system was placed underneath the monitor. The EyeLink was used in monocular mode in the Head Free/Remote setup. A chin rest was used which allows for some head movement but makes sure the head remains in roughly the same position in front of the screen. The remote set up has a lower sample rate than EyeLink's tower mount setup and requires a target sticker placed on the forehead as a reference point. The listed accuracy of the Head Free/Remote setup is also slightly lower; 0.5°while the Head Supported mode is in the range of 0.25°to 0.5°. The accuracy is drop is dealt with in the placement of stimuli as will be discussed in more detail in the next section. The EyeLink system uses a dedicated Host Computer with EyeLink 1000 software for data acquisition.

The EEG recordings were made using the BioSemi ActiveTwo system[4]. The system was used with 32 active electrodes using the extended 10-20 system of electrode placement. Due to using active electrodes, the ActiveTwo system does not require the input impedance to be kept low nor a traditional reference electrode. The system uses a Common Mode Sense (CMS) active electrode and Driven Right Leg (DRL) passive electrode place on the top of the head. For a more detailed description of active electrodes see [37]. BioSemi's own software called ActiView was used for data acquisition and data was recorded with a sample rate of 2048 Hz.

Furthermore four electrodes were used to record an electrooculogram (EOG). Each eye had an electrode placed on the side (between the eye and the temple), and two electrodes were placed above and underneath the left eye. This was specifically the eye that was not tracked to avoid interference with the eye-tracker. The EOG data is gathered as a reference in order to be able to remove eye-movement artefacts from the EEG data.

Syncronisation of the eye-tracker and EEG data is done with markers sent by PsychoPy (see next section) via MIDI. At the start and end of a trial a marker (the trial number) is sent to both the eye-tracker and EEG. These markers can then be used to match the appropriate slices of EEG and eye-tracker data. Direct communication of PsychoPy markers to the EyeLink Host Computer proved not possible. To circumvent this, a separate buffer was made using the FieldTrip Matlab toolbox to which both the eye-tracker data and markers were sent. FieldTrip is a free toolbox developed at the Donders Institute for Brain, Cognition and Behaviour in Nijmegen, Netherlands [38].

## 3.3 Stimuli

The experiment consisted of 30 trials; each trial has 200 stimuli of which 30 are targets and 170 non-targets (distractors). Each trial is preceded by a break and the presentation of the coming trial's target stimulus. The trials were created in Python using PsychoPy, an open-source package for creating

[2]Image source: http://www.cs.jhu.edu/~subodh/457/antialias.html
[3]SR Research Ltd., Ontario, Canada
[4]BioSemi instrumentation, Amsterdam, Netherlands

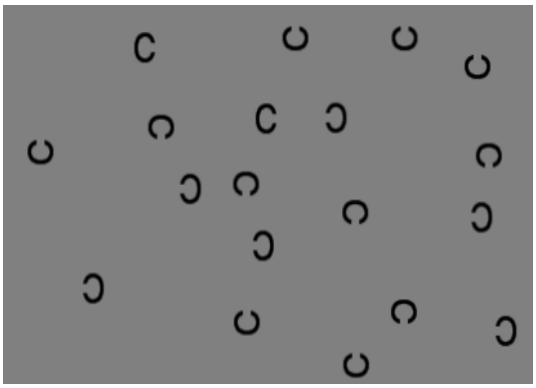
and running experiments [39][40]. No code will be discussed but the full script is provided in Appendix C.

Figure 6a, a section of the stimuli presentation screen, gives an impression of the stimuli that were presented during the experiment (full screen shots of trials are included in Appendix B). The stimuli are 1cm ($\approx 1°$) high c's in the Arial font. The screen displays c's oriented in a random direction, meaning the gap can be on top, bottom, left or right. Each trial consists of 200 such c's of which 30 have that trial's 'target' orientation and the other 170 are distractors which are randomly assigned one of the remaining three orientations. Preceding each trial, the participant is shown a screen with that trial's target orientation for six seconds, as shown in figure 6b.

As shown in figure 6a, stimuli are randomly distributed across the search display. The EyeLink system has a listed Head Free precision of 0.5°, corresponding to 0.53125 cm on the screen given a viewing distance of 60cm. So in order to avoid unclarity over which stimulus was fixated, a margin at least twice that angle should be kept between any two stimuli. The margin between targets was set at 1.2 cm (1.129°). Any measured fixation should then only be associated with one stimulus, easily allowing for the accuracy of the EyeLink. Lastly, there is a border of 2cm from the screen's edge where no stimuli appear. Once participants are done with the trial they can proceed to a break by pressing any mouse button.

A total of 30 trials were generated for the experiment. This was done using a random seed so the same trials are reproduced every run of the experiment. This way each participant is presented the same 30 trials but the trials are presented to each participant in random order.

Lastly, after each trial the participant is presented a screen indicating a break for as long as they want as in figure 6c.

A 

B

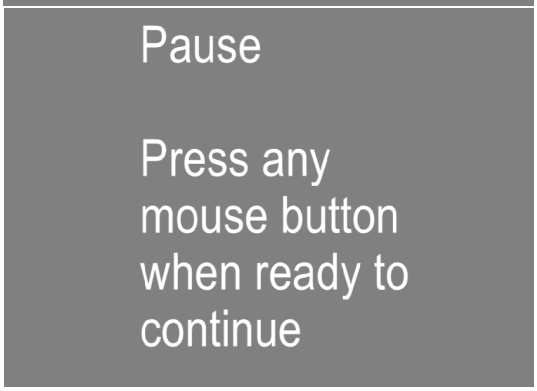

C 


**Figure 6: (A) (section of) the search display, (B) the screen displaying the next trial's target orientation and (C) the screen telling the participant to press a mouse button to continue to the next trial.**

### 3.4 Task Design and Procedure

Before the start of the experiment all participants were informed about the procedure and asked to sign an informed consent form.

Participants are seated comfortably in front of the monitor with the chin rest set at a distance of 60cm from the screen. The EEG cap and EOG electrodes are set up and the EyeLink is calibrated.

The participants received the following task instructions; They were told the experiment consists of 30 blocks of target presentation, stimuli and a break and each stimuli presentation contains 30 targets. The target presentation shows a large c with that block's target orientation for five seconds and after that the stimuli appear. Participants are instructed to find all stimuli with the target orientation and proceed by clicking any mouse button when they think they have found all the targets. They were told the breaks last as long as they like and that they can proceed to the next block by clicking any mouse button and that during the breaks they may leave the chin

rest. As seen in figure 6 the experiment contains text as a reminder of the instructions, but participants can make contact through an intercom for questions at any time.

Lastly, the participants were instructed to try and keep movements of the head to a minimum and not to touch their forehead during the breaks. This is to prevent them from accidentally rubbing off the eye-tracker's reference sticker.

# 4 Analysis

## 4.1 Fixation Processing

Fixations were determined using the default software that comes with the EyeLink system. The system collects the fixation events which occur when gaze position remains steady for at least 40 ms. The EyeLink then saves these events as; time of fixation start and end, average x and y position in pixels and the trial number. During blinks the EyeLink loses track of the pupill meaning these episodes will not qualify as a fixation.

All EEG and fixation processing, classification and statistical testing was done using Matlab 2015a[5] and the Fieltrip Toolbox for Matlab [36].

The fixation positions were compared to the locations of the targets and non-targets to determine target fixations. A fixation was marked as a target fixation if the euclidean distance between the fixation and the center of a target was smaller than 14 pixels (0.55cm), non-target fixations were marked in the same way. Although the listed accuracy of the EyeLink is 0.53 cm, all stimuli were placed at least 1.2 cm apart and all fixations were checked to make sure they were not coupled to more than one stimulus. None of the fixations were found to be coupled to multiple stimuli so it was safe to use this cut-off point.

Lastly the target fixations were marked as cancellations or re-cancellations. A fixation was marked as re-cancellation if a fixation on the same target but with an earlier fixation onset time exists, and marked as a cancellation otherwise.

## 4.2 EEG Pre-processing

Pre-processing started by slicing the EEG data into trials using the markers sent by PsychoPy. This results in 30 trials per participant, although one participant skipped a trial due to missing the target presentation. That trial was removed from the analysis. During slicing the data was downsampled to 256 Hz.

Next the common average reference (CAR) was subtracted from the signal. This was done by computing the average over all electrodes for each frame and subtracting it from the data. Re-referencing is a common technique to remove common signals such as noise from the data such that only the signals unique to each electrode remain, the signal that is left is then relative to a base line or zero level.

Outlying epochs were removed from the data by rejecting those with a variance that differed more than 3.5 standard deviations from the median.

Linear trends (changes in the DC offset) are removed as small changes in conductance between the scalp and the electrodes occur over the duration of the experiment which results in the signal deviating from the baseline. The result is a difference in average amplitude of the signal and thus differences in amplitude of the FRPs at different stages of the experiment. Linear trends in the data are removed by fitting the signal to a curve and removing that curve from the data thereby centering all data around the same baseline.

The EOG data is used to regress the eye-movement artifacts out of the EEG channels and lastly a 0.5 Hz High pass filter and a 12 Hz low pass are applied to remove frequencies that are not of interest for the analysis.

The EEG data is then sliced into FRPs using the onset of the fixations. A time window from 0 to 500 ms around the fixation onset was used to slice the EEG. All fixations shorter than the time window of 500 ms were rejected as they contain artefacts such as the next FRP and eye-movement. This time window should contain even relatively late components of the FRPs such as the P300, and is a compromise between including more features and having to reject to many samples as only a few fixations lasted longer than 500 ms. This selection brought the total number of cancellations to 232 and re-cancellations to 86.

## 4.3 Classification

To investigate whether there is a difference between cancellations and re-cancellations a classifier is then trained on the data. Given samples from both classes a classifier can be trained to distinguish between these classes, in this case cancellation and re-cancellation

[5]The MathWorks, Inc., Natick, Massachusetts, USA

and target and non-target, based on features in the FRPs. The features here are the amplitudes of the brain activity as measured by the EEG. If a classifier can distinguish between the two classes, this is a clear indication that samples of the two classes generally display different features from each other.

The classifier was trained using 5 fold cross validation. In cross validation the data is partitioned into (in this case 5) subsets, the folds are rotated between being training and test data where each fold is used as training data once. Before classification the data is downsampled to 32 Hz so for 500ms of EEG data there are 16 features per channel, or 512 features in total for the classifier.

The classes in the training data need to be balanced such that there is an equal amount of FRPs from both classes to train on. If the classes are not equally distributed the classifier will have the tendency to prefer the larger class as this would lead to a lower error, or higher classification result.

The eye-tracker data was analysed to see if there are significant differences in the duration of fixations for cancellations/re-cancellations and targets/non-targets. The fixations were analysed using analyses of variance (ANOVA). For this analysis it was not necessary to reject fixations based on fixation length, so all the fixation data was included. Only outliers were removed based on a distance of 3.5 times standard deviation from the mean.

The fixation duration was then used to classify fixations using a binary decision tree, in order to compare the classification performance to that of the classifier based on the EEG data. The classifier finds the optimal cut-off point in fixation duration, where all fixations under the cut-off are designated as one class and all those above the cut-off as the other class. For the classification samples of the largest class were again randomly selected to make sure that both classes were equally represented in the training data.

# 5 Results

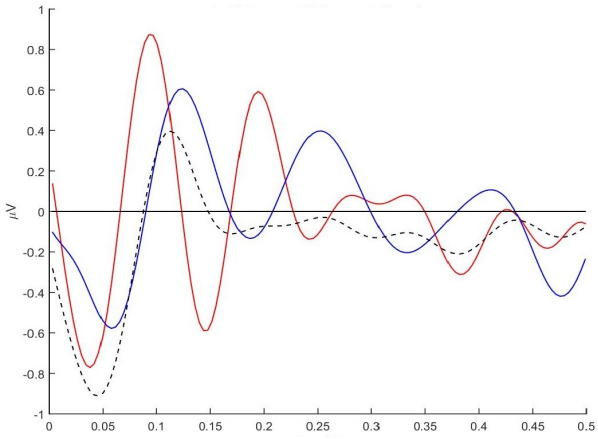
Table 1 shows the number of fixations that were included in the EEG analysis.

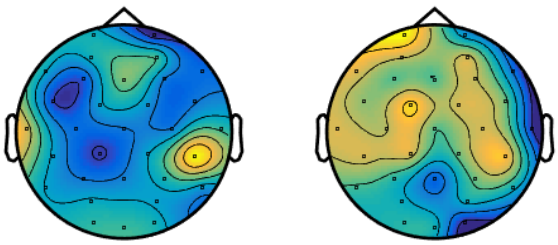
| Participant | Cancellations | Re-cancellations | non-target fixations |
|---|---|---|---|
| 1 | 57 | 35 | 1600 |
| 2 | 75 | 23 | 649 |
| 3 | 61 | 16 | 556 |
| 4 | 39 | 12 | 1541 |
| Average | 58 | 21.5 | 1086.5 |

**Table 1: the total number of fixations valid for analysis**

The grand average FRPs per class are displayed in figure 7a. This is the average over all participants where each participant received equal weight. Only the channel Cz is displayed here for legibility. Grand average plots for all 32 channels are included in appendix A. Figure 7b shows the topographical maps obtained for both class distinctions between 0.35 and 0.45 seconds after fixation. This time frame is chosen as it should contain the P300.

A



B

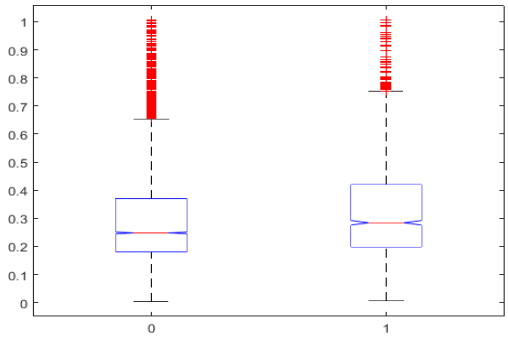
**figure 7:(A) Plot of grand average FRPs at Cz, the red line is cancellation, blue is re-cancellation and the dotted line is non-target FRP. (B) Topographical map showing the contrast between classes at the time interval of the P300 response.**

Classification performance per participant is shown in table 2. This is the average over all 5 folds of the percentage of FRPs that are correctly classified in the test set. One participant scored well above chance level (59%), two scored just barely above chance level (53%) and one participant did not score better than chance at 50%. This indicates that the FRPs cannot be used to correctly classify the FRPs for three out of four patients.

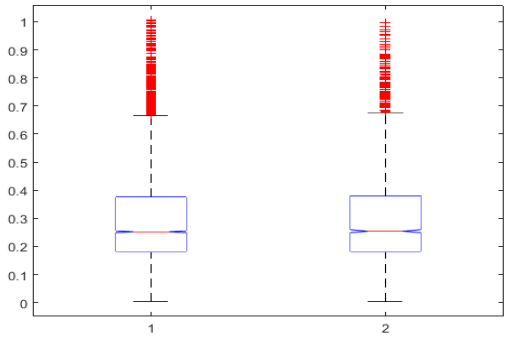
| Participant | Cancellations vs re-cancellations | targets vs non-targets |
|---|---|---|
| 1 | 50% | 50% |
| 2 | 53% | 53% |
| 3 | 59% | 57% |
| 4 | 53% | 54% |
| Average | 53.75% | 53.5% |

**Table 2: classification score per participant**

The fixation data was first checked for normality of the distribution using a Kolmogorov-Smirnov test. This tests the null hypothesis that the data is normally distributed against the alternative hypothesis that it is not. The test was not significant (p= 0.317) which means the data comes from a standard normal distribution. The ANOVA test results showed a significant difference in mean length for targets versus non-targets fixations (mean=0.291 and 0.329 s, F=90.968, p<0.000), but not for first versus subsequent fixations (mean=0.296 and 0.297 s, F=0.0175, p=0.8947). figure 8 displays the box plots of the fixation length distribution for both tests.




A




B

**Figure 8: Box-plots for the distribution of fixation length (A) 0 = non target, 1 = target. (B) 1 = first fixation, 2 = subsequent fixation.**

The binary decision tree (non-target < 0.278 s <= target) based on the fixation length had a classification rate of 54,3% for targets versus non targets, slighty higher than the EEG classifier for three out of four participants. The tree for first fixation versus subsequent fixation (first fixation < 0.192 s<= subsequent fixation) had a classification rate of 53.4% which is around the same performance as the EEG data for three out of four participants.

# 6 Discussion

The current study aimed to examine and compare the FRPs in cancellations (first fixations) and those in re-cancellations (subsequent fixations) using a combination of EEG and eye-tracking. In order to see if there is a difference between cancellations and re-cancellation, participants were recorded while performing a task involving SWM. In this section I will discuss possible limitations and suggest several solutions to these limitations.

## 6.1 Possible Limitations

The result is inconclusive as although on visual inspection the grand average FRPs for these events seem distinct, a classifier could not label the FRPs well above chance level, as the average classification score is 53.75% which is barely above chance level. So even though the average FRPs seem distinct upon visual inspection, individual responses did not seem to display any significant differences for three out of four participants. One participant had a classification score of 59% however, which provides some support for the difference in FRPs. A study using a simi-

lar set-up got an average 62% classification rate for targets versus non-targets [20]. The difference between targets and non-targets is proven by research so similar classification results for one participant seems promising (see [41] for a review on the P300 response). Even more so as this participant's score on the cancellation versus re-cancellation classification is slightly higher than that for target versus non-target.

One explanation for the low overall classification results might simply be the low number of samples per class. The highest number of cancellations for any participant was 75 and two participants had less than 20 re-cancellations. It was considered to drop such participants from the analysis, however due to the low number of participants it was not deemed beneficial to lose more data. When the number of FRPs is low, exceptional spikes have a larger affect on the data. Upon visual inspection of the grand averages (figure 7) this becomes clear as especially the smallest class (re-cancellations) tend to show more and higher spikes. On the contrary the largest class (non-target) is an average of over 4000 FRPs and it clearly shows less extreme values. It was also observed that participants made relatively little re-cancellations which is to be expected of healthy subjects [14][42]. Future research should improve on this task design to increase the number of re-cancellations, as a high number of targets alone did not evoke enough re-cancellations.

Another explanation is the large number of c's that are displayed in a small space under the current set-up. As a result, both targets and non-targets could be seen in the periphery of a participant's vision. This has two possible effects: first, the number of total fixations is reduced as participants can process multiple stimuli in one fixation, and second, FRPs that are now assumed to be a response to the fixated stimulus might be a response to another or even multiple stimuli.

The latter effect seems the most serious as this reduces the difference between cancellations and re-cancellations as examples of both classes enter each others sample set. When a participant fixates a target for the second time, but in his periphery notices a new target, the FRP that enters the re-cancellation set might actually be a cancellation FRP. And when the participant fixates on a target that was already seen in the periphery the opposite happens, as the event is labelled as a cancellation, while it is actually a re-cancellation FRP. Furthermore, as attentional resources get engaged by multiple stimuli, smaller P300 amplitudes are produced, thus reducing the quality of the FRPs [41].

In the same vein, participants might make mistakes in deciding if a target is new or not. 30 targets is a large number to retain in SWM and this could also lead to old targets being perceived as new by the participant but labelled as a re-cancellation and vice versa. Usually, a button box or similar device is used so participants can indicate whether or not they think a target is new[19]. This way erroneous trials can be removed from the analysis. This was not done in the current research in order to avoid muscle movement artifacts but this decision came with its own limitations.

How can the current results be explained if the cause was not bad quality data as explained above? This could mean there might simply be no difference between cancellations and re-cancellations. The test on the fixation duration showed that there is no significant difference in mean fixation length between cancellations and re-cancellations as well (p=0.8947), and seems to support this interpretation of the results.

Evidence against this is that the fact the experiment also failed to confirm the differences between target and non-target fixations. Although a the ANOVA showed significant differences in fixation length between targets and non-target ($p < 1.74^{-21}$), the classifier could not label these classes well above chance level (average classification result 53.5%). The difference in average P300 amplitude between target and non-target stimuli is a well documented phenomenon [41][43]. The absence of outspoken differences between target and non-target FRPs for three out of four participants seems to confirm that the data set was either to small, of bad quality or a combination of both.

## 6.2 Suggested Improvements

Based on the results of the current study, improvements to the experiment were made that address some of the limitations of the current study. Since the current study shows some promise of differences between cancellation and re-cancellation FRPs, the improved experiment is still aimed at examining SWM using EEG and eye-tracking. The improvements are focused on the challenges that were found in collecting the data, namely; gathering more samples per class and reducing noise introduced during the experiment.

The first limitation was that there were not enough FRPs especially for the re-cancellation class.

This is a challenge in terms of formulating the task; as you can not simply ask the participants to make some re-cancellations these have to occur naturally due to task requirements.

The task was changed to one where the participants have to find and remember ten target locations in between 30 non-targets. When they indicate by mouse-click that they found all targets the trial proceeds, and the same screen targets appear but one non-target is now changed into a target. Participants are instructed to find all eleven targets now decide which ones they already saw and click on the new target. Participants are instructed to find all eleven targets and then decide which target is new, and so creating at least ten re-cancellations every trial.

Ten locations is also a more realistic number of targets to retain in SWM and should lead to less mistakes from participants. This might solve the problem of erroneous trials without requiring a button box or other feedback device. At the end of each trial the participant indicate which target is the new one by mouse click and trials with a mistake could simply be removed based on mouse position. This way muscle movement artifacts are consistently present only during the last fixation in every trial and those can simply be removed with minimal loss to the data.

The last improvement solves the problem of stimuli processing from the peripheral vision. Targets are masked by placing a grid of o's around them. This makes it harder to discern the orientation of the c's opening from the peripheral vision and requires participants to gaze at the center of the stimuli. This might also have to positive side-effect of reducing eye-tracker errors as the masking serves as a fixation aid. Figure 8 shows an example of the masked stimuli. A full screenshot of a trial is included in Appendix B and illustrates that it is difficult to discern the orientation of the stimuli in the periphery of vision when you are focussing on the middle of a stimulus.

Lastly, fixation length can be increased by simply instructing participants to gaze at the center of the stimuli and keep their fixation there for about a second. This does not inform the participant about the experiments goal of comparing cancellation and re-cancellation FRPs nor force any of these responses.

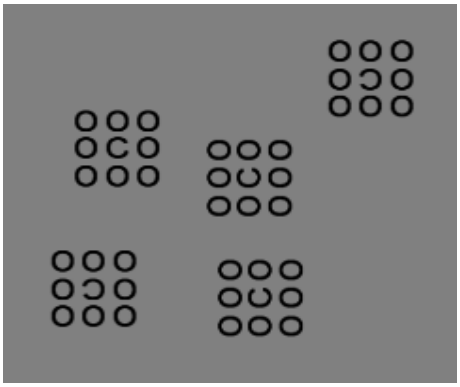


**figure 8: example of the masked stimuli**

# 7 Conclusion

The conclusions of this study are fourfold:
- Cancellations could be distinguished from re-cancellations for one out of four participants.
- Mean fixation length was not significantly different between cancellations and re-cancellations, it was for targets and non-targets however.
- Low overal results are likely due to a small data set and low quality FRPs.
- The current experiment can be improved to tackle the challenges in data collection.

The recommendation based on the current results would be to re-run the experiment using the same equipment but with an improved task design. One participant showed promise of differences between cancellations and re-cancellations and the improved experimental design could lead to the same results in others. The full code for both the old and new experiment is included in Appendix D.

# 8 Acknowledgements

I would like to thank my thesis supervisors Dr. Ervin Poljac and Marjolein van der Waal for their guidance, advice and feedback, this thesis is all the better for it. I would also like to thank Dr. Philip van den Broek for his support in the lab, and Dr. Jason Farquhar for looking at my analysis. Lastly I want to thank my participants, who gave their time to help a friend write his thesis, and whose insightful comments on the experiment were a great help in formulating the improvements to the task paradigm.

# 9 Literature


[1] Karnath, H. O., Berger, M. F., Kker, W., & Rorden, C. (2004). The anatomy of spatial neglect based on voxelwise statistical analysis: A study of 140 patients. *Cerebral Cortex, 14(October),* 1164-1172. doi:10.1093/cercor/bhh076

[2] Buxbaum, L. J., Ferraro, M. K., Veramonti, T., Farne, a, Whyte, J., Ladavas, E., Frassinetti, F., & Coslett, H. B. (2004). Hemispatial neglect: Subtypes, neuroanatomy, and disability. *Neurology, 62,* 749-756. doi:10.1212/01.WNL.0000113730.73031.F4

[3] Ringman, J. M., Saver, J. L., Woolson, R. F., Clarke, W. R., & Adams, H. P. (2004). Frequency, risk factors, anatomy, and course of unilateral neglect in an acute stroke cohort. *Neurology, 63,* 468-474. doi:10.1212/01.WNL.0000133011.10689.CE

[4] Heilman K. M., Valenstein E., & Watson R. T. (2000). Neglect and related disorders. *Seminars in Neurology, 20(4),* 463-470.

[5] Bowen A, & Lincoln N. B. (2007). Cognitive rehabilitation for spatial neglect following stroke. *Cochrane Database of Systematic Reviews 2007, Issue 2.* Art. No.: CD003586. DOI: 10.1002/14651858.CD003586.pub2.

[6] Fasotti, L., & van Kessel, M. (2013). Novel insights in the rehabilitation of neglect. *Frontiers in Human Neuroscience, 7(November),* 780. doi:10.3389/fnhum.2013.00780

[7] Corbetta, M. (2014). Hemispatial Neglect : Clinic , Pathogenesis , and Treatment. *Seminars in Neurology, 5(34),* 514-523.

[8] Vossel, S., Weiss, P. H., Eschenbeck, P., & Fink, G. R. (2013). Anosognosia, neglect, extinction and lesion site predict impairment of daily living after right-hemispheric stroke. *Cortex, 49(7),* 1782-1789. doi:10.1016/j.cortex.2012.12.011

[9] Angeli, V., Benassi, M. G., & Ladavas, E. (2004). Recovery of oculo-motor bias in neglect patients after prism adaptation. *Neuropsychologia, 42,* 1223-1234. doi:10.1016/ j.neuropsychologia.2004.01.007

[10] Barrett, A. M., Goedert K.M., & Basso, J.C. (2012). Prism adaptation for spatial neglect after stroke: Translational practice gaps. *Nat Rev Neurol, 8(10),* 567-577. doi: 10.1038/nrneurol.2012.170

[11] Lincoln, N. B., Drummond, A., & Walker, M. F. (1995) The treatment of visual neglect using feedback of eye-movements - A pilot study. Disability and Rehabilitation, 17(8), 413-417.

[12] Husain, M., Shapiro, K.,Martin, J., & Kennard, C. (1997). Abnormal temporal dynamics of visual attention in spatial neglect patients. *Nature 385,* 154-156. doi:10.1038/385154a0

[13] Danckert J., & Ferber, S. (2006) Revisiting unilateral neglect. *Neuropsychologia, 44,* 987-1006. doi:10.1016/ j.neuropsychologia.2005.09.004

[14] Wansard, M., Meulemans, T., Gillet, S., Segovia, F., Bastin, C., Toba, M. N., & Bartolomeo, P. (2014). Visual neglect: Is there a relationship between impaired spatial working memory and re-cancellation? *Experimental Brain Research, 232,* 3333-3343. doi:10.1007/s00221-014-4028-4

[15] Striemer, C. L., Ferber, S.,& Danckert, J. (2013). Spatial working memory deficits represent a core challenge for rehabilitating neglect. *Frontiers in Human Neuroscience, 7(June),* article 334. doi:10.3389/ fnhum.2013.00334

[16] Deouell, L. Y., Hämäläinen, H., & Bentin, S. (2000). Unilateral neglect after right-hemisphere damage: contributions from event-related potentials. *Audiology & Neuro-Otology, 5,* 225-234. doi:10.1159/000013884

[17] Awh, E., & Jonides, J. (2001). Overlapping mechanisms of attention and spatial working memory. *Trends in Cognitive Sciences, 5(3),* 119-126.

[18] Husain, M., Mannan, S., Hodgson, T., Wojciulik, E., Driver, J., & Kennard, C. (2001). Impaired spatial working memory across saccades contributes to abnormal search in parietal neglect. *Brain, 124,* 941-952.

[19] Saevarsson, S., Kristjánsson, Á., Bach, M., & Heinrich, S. P. (2012). P300 in neglect. *Clinical Neurophysiology, 123,* 496506. doi:10.1016/j.clinph.2011.07.028

[20] Brouwer, A. M., Reuderink, B., Vincent, J., van Gerven, M. A. J., & van Erp, J. B. F. (2013). Distinguishing between target and nontarget fixations in a visual search task using fixation-related potentials. *Journal of Vision, 13(3): 17,* 1-10. doi:10.1167/13.3.17

[21] Putrino, D. (2014). Telerehabilitation and emerging virtual reality approaches to stroke rehabilitation. *Current Opinion in Neurology, 27,* 631-636. doi:10.1097/WCO.0000000000000152

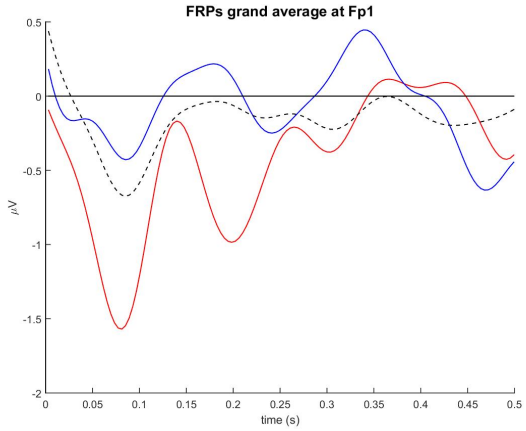
[22] Barrett, a. M., & Muzaffar, T. (2014). Spatial cognitive rehabilitation and motor recovery after stroke. *Current Opinion in Neurology, 27,* 653-658. doi:10.1097/WCO.0000000000000148
[23] Kolb, W. & Whishaw, I. Q. (2011). An Introduction to Brain and Behaviour. Worth Publishers, New York.
[24] Fredette, M. (n.d.). Journal of the Association for Information Precision is in the Eye of the Beholder : Application of Eye Fixation-Related Potentials to Information Systems Research Precision is in the Eye of the Beholder : Application of Eye Fixation-Related Potentials to, 15(April 2013), 651678.
[25] Picton, T. W. (1992). The P300 Wave of the Human Event-Related Potential. *Journal of Clinical Neurophysiology, 9(4),* 456-479.
[26] Guan, Cuntai., Thulasidas, M. & Wu, J. (2004). High Performance P300 Speller for Brain-Computer Interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 14(2),* 221-224
[27] Körner, C., Braunstein, V., Stangl, M., Schlögl, A., Neuper, C., & Ischebeck, A. (2014). Sequential effects in continued visual search: Using fixation-related potentials to compare distractor processing before and after target detection. *Psychophysiology, 51,* 385395. doi:10.1111/psyp.12062
[28] Kaunitz, L. N., Kamienkowski, J. E., Varatharajah, A., Sigman, M., Quiroga, R. Q., & Ison, M. J. (2014). Looking for a face in the crowd: Fixation-related potentials in an eye-movement visual search task. *NeuroImage, 89,* 297305. doi:10.1016/j.neuroimage.2013.12.006
[29] Malmivuo, J., & Plonsey, R. (1995). Bioelectromagnetism - Principles and Applications of Bioelectric and Biomagnetic Fields. Chapter 28: The electric signals originating in the eye. Oxford University Press, New York.
[30] Woestenburg, J.C., Verbaten, M. N., & Slangen, J. L. (1983). The removal of the eye-movement artifact from the EEG by regression analisys in the frequency domain. *Biological Psychology, 16(1-2),* 127-147. doi:10.1016/0301-0511(83)90059-5
[31] Urigüen, J. A., & Garcia-Zapirian, B. (2015). EEG artifact removal- state-of-the-art and guidelines. *Journal of Neural Engineering, 12(April).* doi:10.1088/1741-2560/12/3/031001
[32] Gutiérrez-Garralda, J. M., Hernandez-Castillo, C. R., Barrios, F. A., Pasaye, E. H., & Fernandez-Ruiz, J. (2014). Neural correlates of spatial working memory manipulation in a sequential Vernier discrimination task. *NeuroReport, 25,* 14181423. doi:10.1097/WNR.0000000000000280
[33] Wojciulik, E., Husain, M., Clarke, K., Driver, J., (2001) Spatial working memory deficit in unilateral neglect. *Neuropsychologia , 39 (4 )* pp.390 -396 . 10.1016/S0028-3932(00)00131-7.
[34] Tatum, W. O., Dworetzky, B. A., & Schomer, D. L. (2011). Artifact and recording concepts in EEG. *Journal of Clinical Neurophysiology, 28(3),* 252-263.
[35] Marella, S. (2012, November 14). EEG artifacts. Retrieved June 15, 2015, from http://www.slideshare.net/SudhakarMarella/eeg-artifacts-15175461
[36] Repovš, G. (2010). Dealing with noise in EEG recording and data analysis. *Infor Med Slov, 15(1),* 18-25.
[37] MettingVanRijn, A. C., Kuiper, A. P., Dankers T. E., & Grimbergen C. A. (1996) Low-cost Active Electrode Improves the Resolution in Biopotential Recordings. *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE, 1,* 101-102. doi: 10.1109/IEMBS.1996.656866
[38] Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J.(2011) FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data. *Computational Intelligence and Neuroscience, vol. 2011,* Article ID 156869. doi:10.1155/2011/156869
[39] Peirce J. W. (2009) Generating stimuli for neuroscience using PsychoPy. *Front. Neuroinform., 2:10.* doi:10.3389/neuro.11.010.2008
[40] Peirce, J. W. (2007) PsychoPy - Psychophysics software in Python. *Journal of Neuroscience Methods, 162(1-2),* 8-13.
[41] Polich, J. (2007). Updating P300: An integrative theory of P3a and P3b. *Clinical Neurophysiology, 118,* 2128-2148.
[42] Mannan, S. K., Mort, D. J., Hodgson, T. L., Driver, J., Kennard, C., & Husain, M. (2005). Revisiting Previously Searched Locations in Visual Neglect : Role of Right Parietal and Frontal Lesions in Misjudging

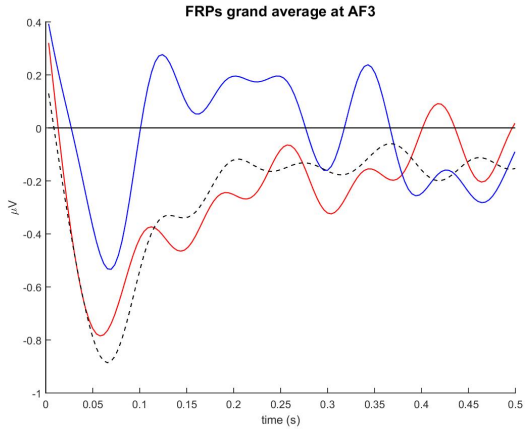
Old Locations as New. *Journal of Cognitive Neuroscience, 17:2,* 340354.
[43] Machado, S., Arias-carrión, O., Sampaio, I., Bittencourt, J., Velasques, B., Teixeira, S., & Ribeiro, P. (2014). Source imaging of P300 visual evoked potentials and cognitive functions in healthy subjects. *Clinical EEG and Neuroscience. 45(2),* 262-268. doi:10.1177/1550059413514389

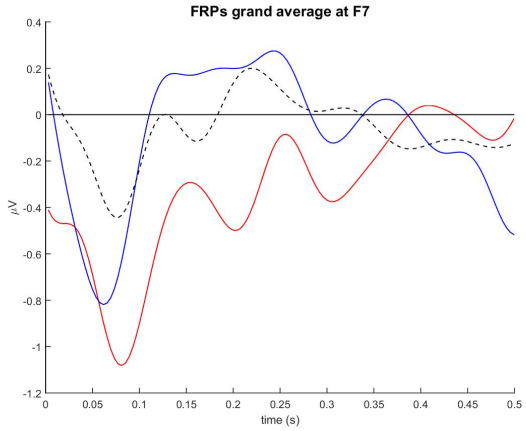
# 10 Appendices

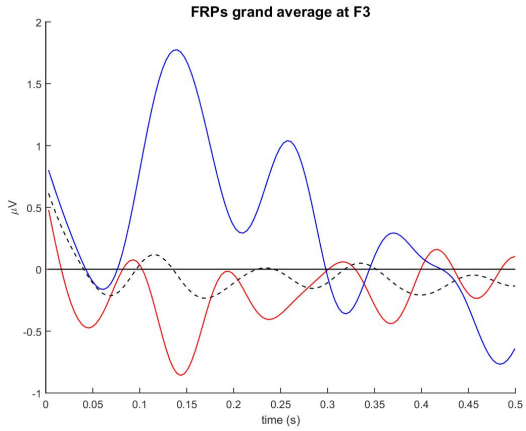
## 10.1 Appendix A

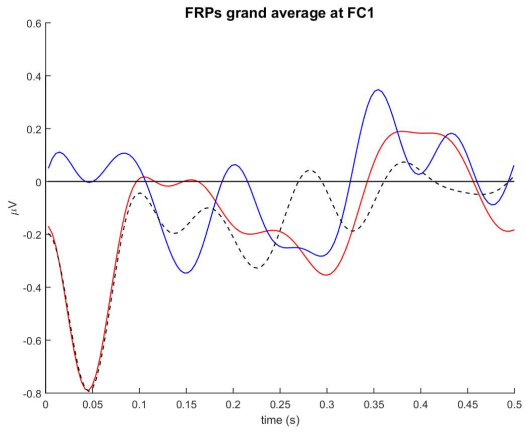
Appendix A includes the grand average plots for all recorded locations. Legend: red - cancellation, blue - re-cancellation, dotted - non-target

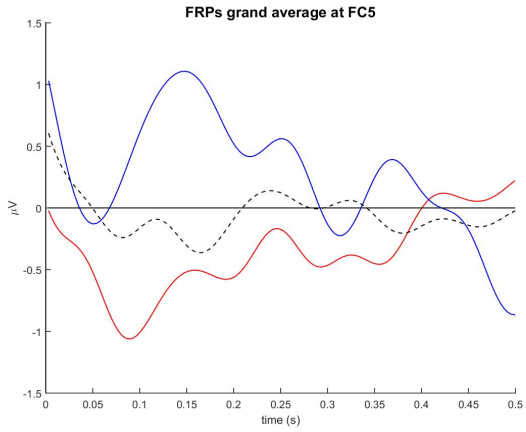


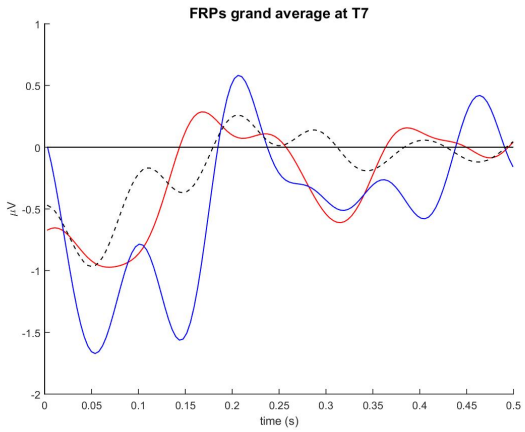



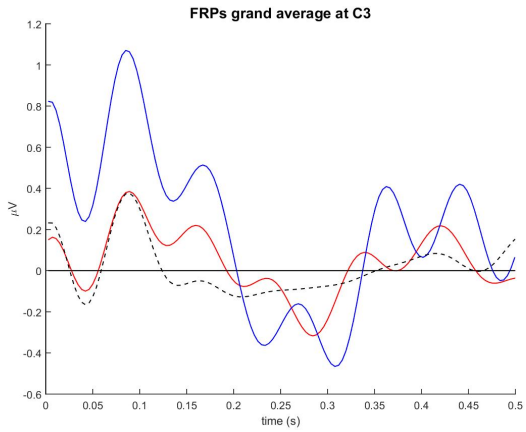



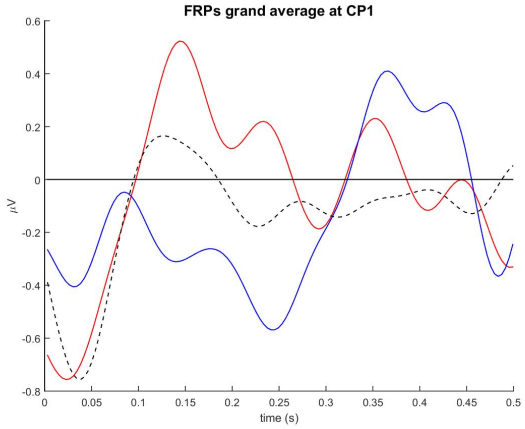



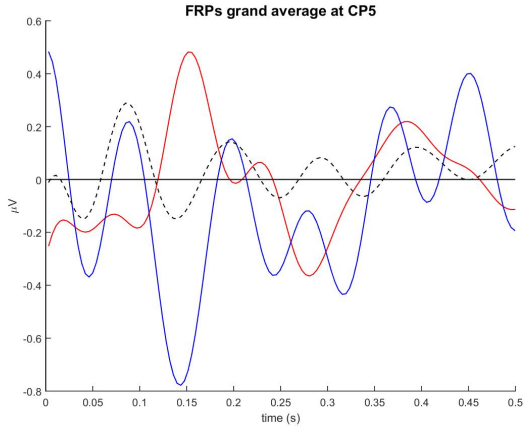



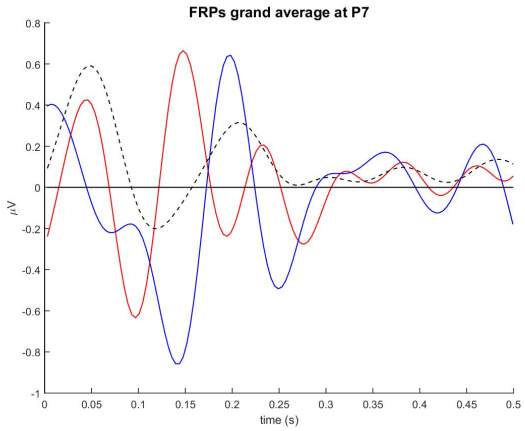



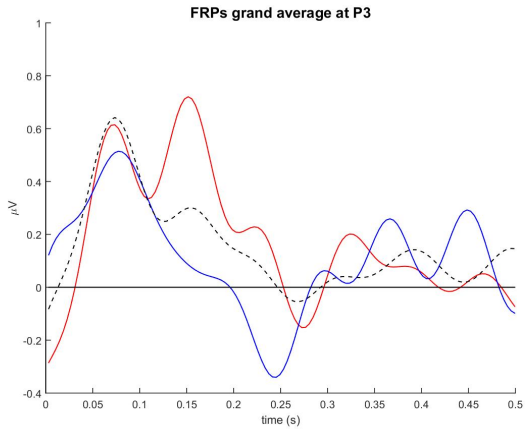

FRPs grand average at T7
μV
time (s)

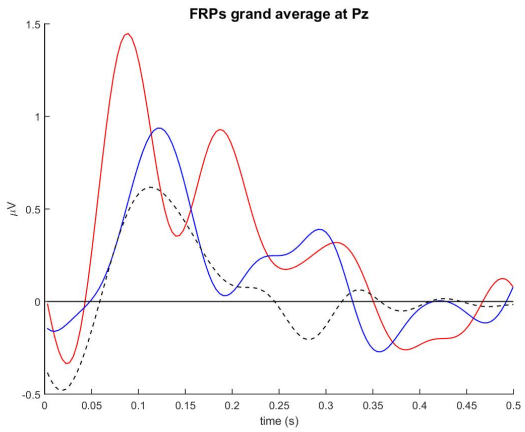

FRPs grand average at C3
μV
time (s)

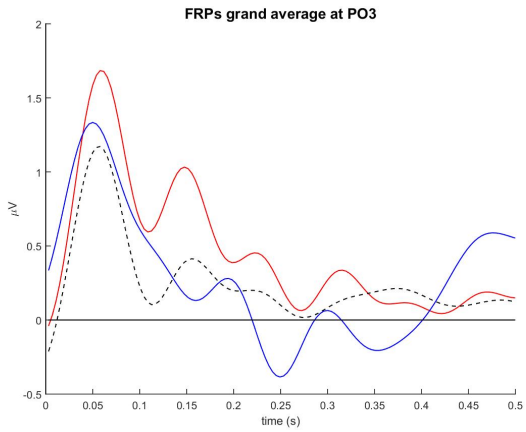

FRPs grand average at CP1
μV
time (s)

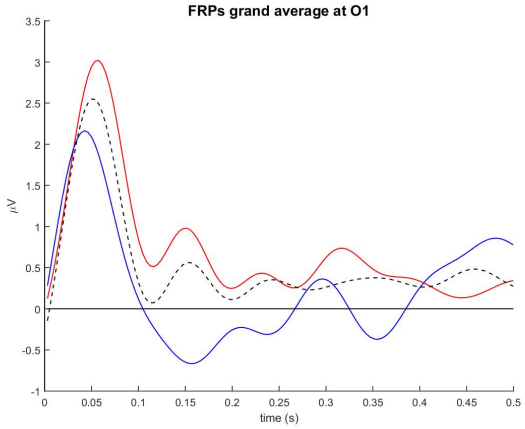

FRPs grand average at CP5
μV
time (s)

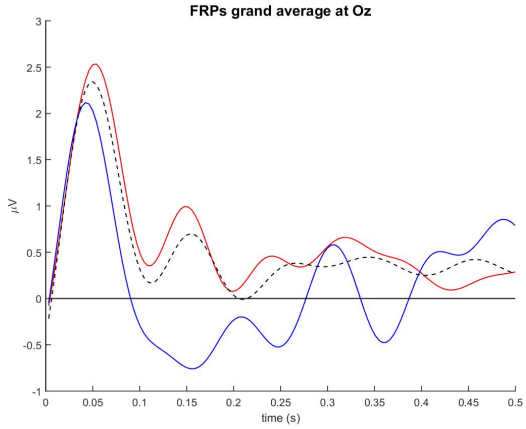

FRPs grand average at P7
μV
time (s)

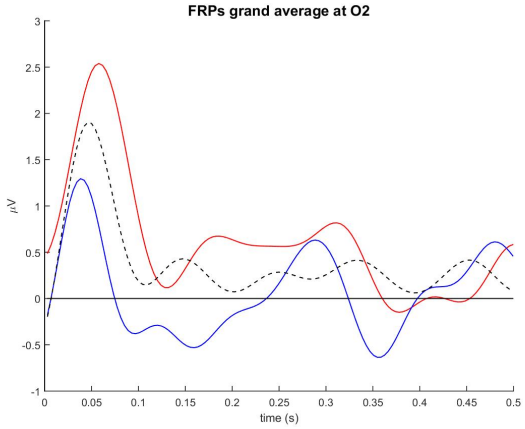

FRPs grand average at P3
μV
time (s)

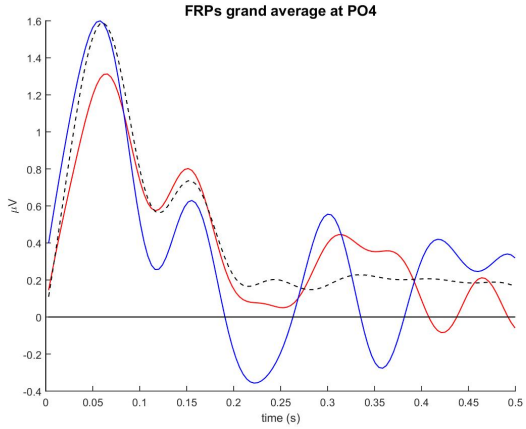

FRPs grand average at Pz
μV
time (s)

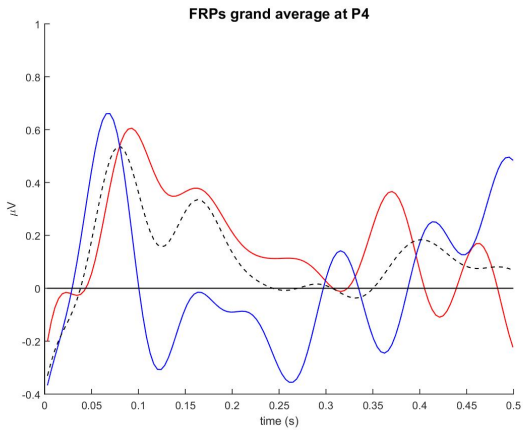

FRPs grand average at PO3
μV
time (s)

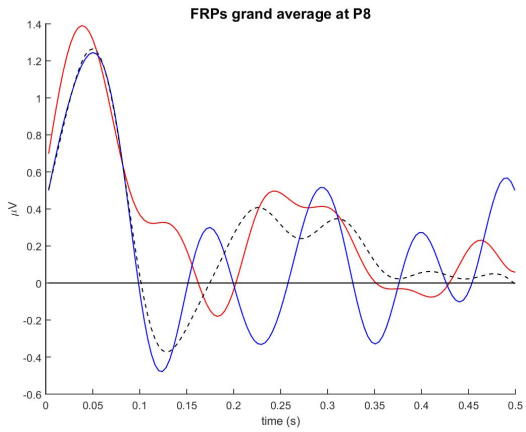

FRPs grand average at O1
μV
time (s)

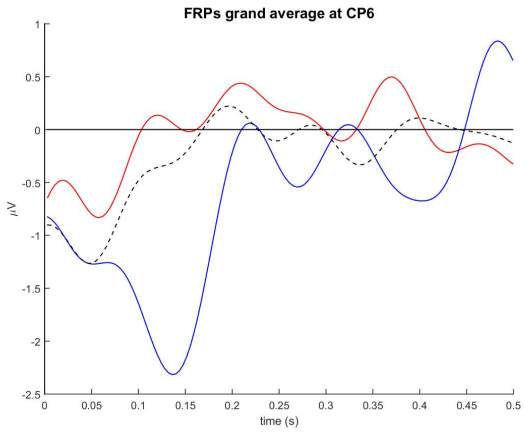

FRPs grand average at Oz
μV
time (s)

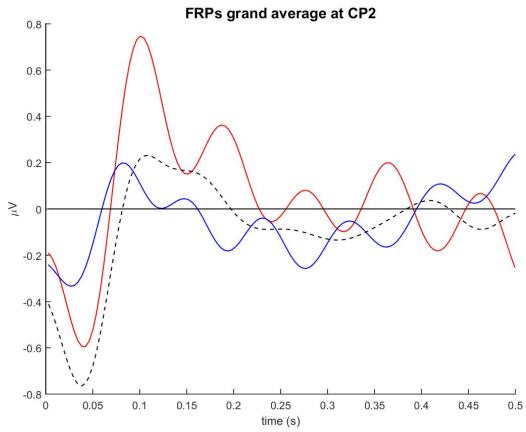

FRPs grand average at O2
μV
time (s)

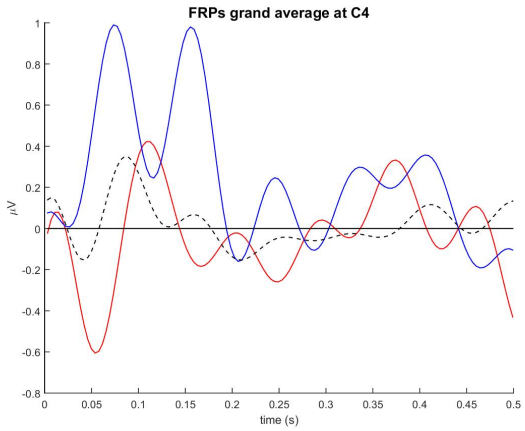

FRPs grand average at PO4
μV
time (s)

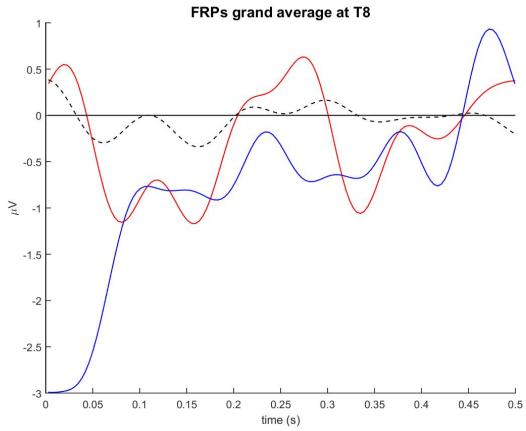

FRPs grand average at P4
μV
time (s)

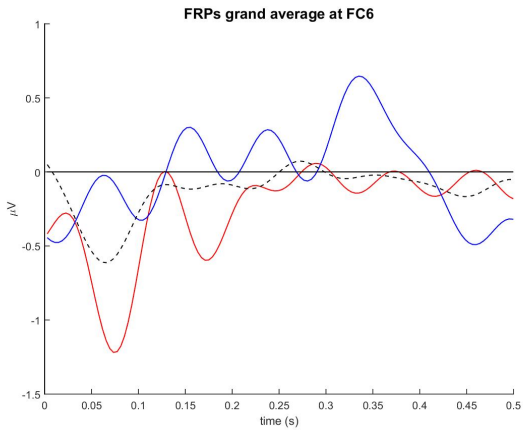

FRPs grand average at P8
μV
time (s)

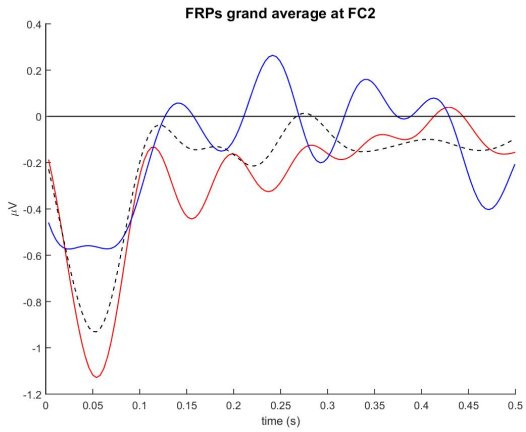

FRPs grand average at CP6
μV
time (s)

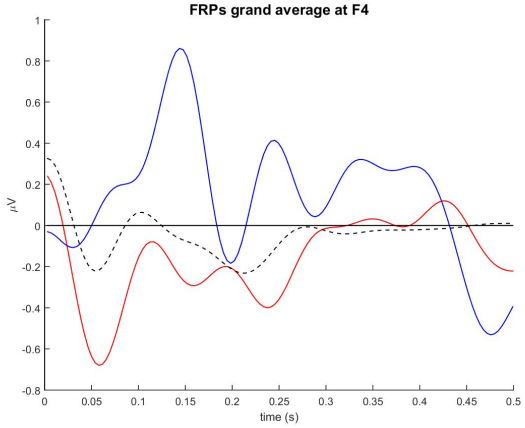

FRPs grand average at CP2
μV
time (s)

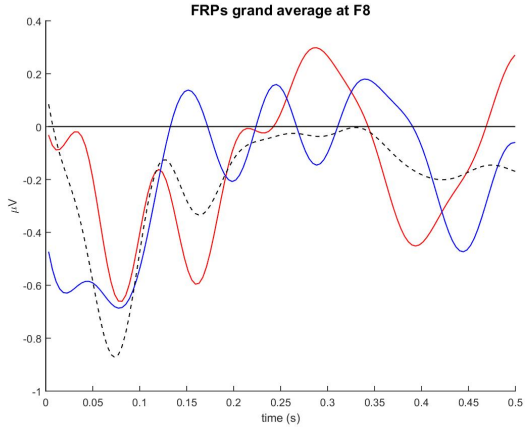

FRPs grand average at C4
μV
time (s)

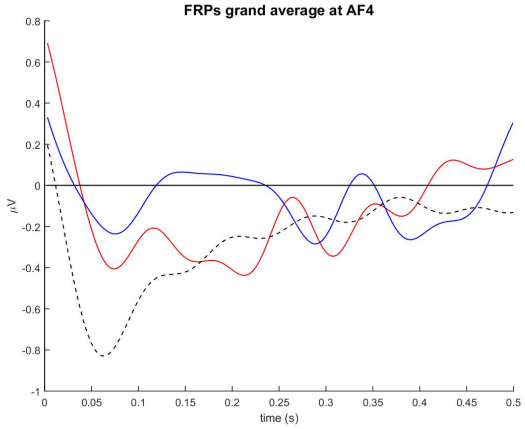

FRPs grand average at T8
μV
time (s)

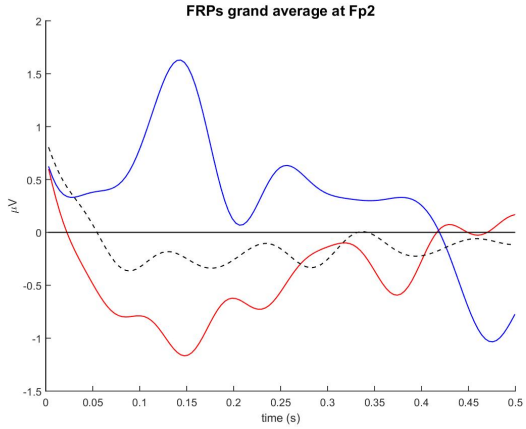

FRPs grand average at FC6
μV
time (s)

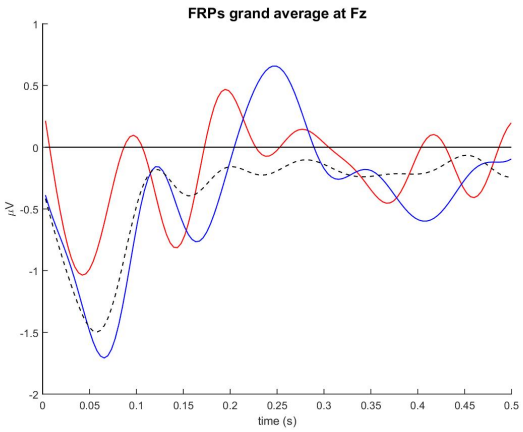

FRPs grand average at FC2
μV
time (s)

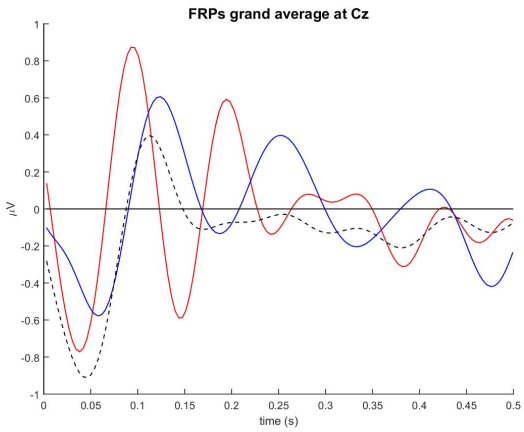

FRPs grand average at F4
μV
time (s)


FRPs grand average at F8
μV
time (s)


FRPs grand average at AF4
μV
time (s)

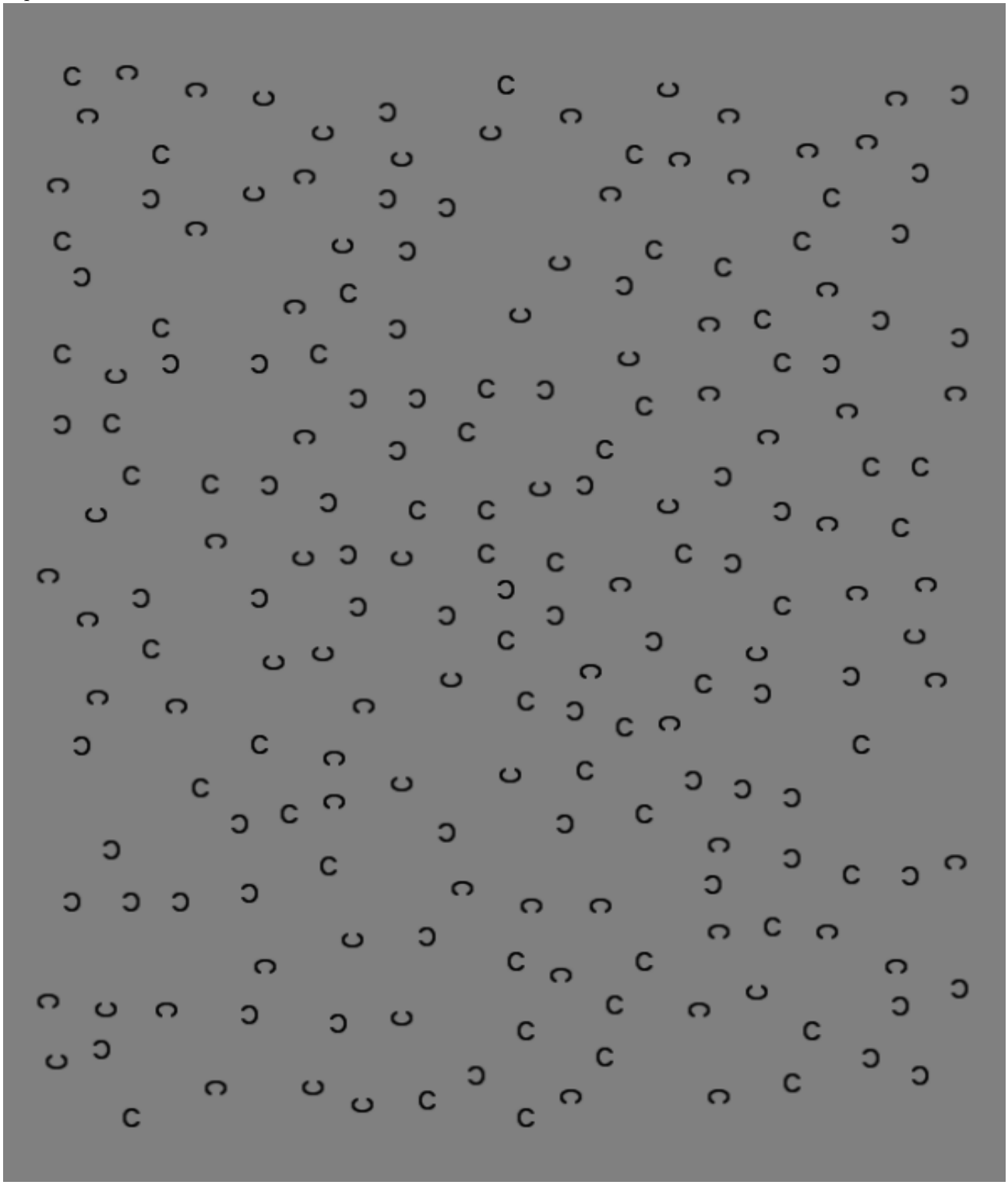

FRPs grand average at Fp2
μV
time (s)

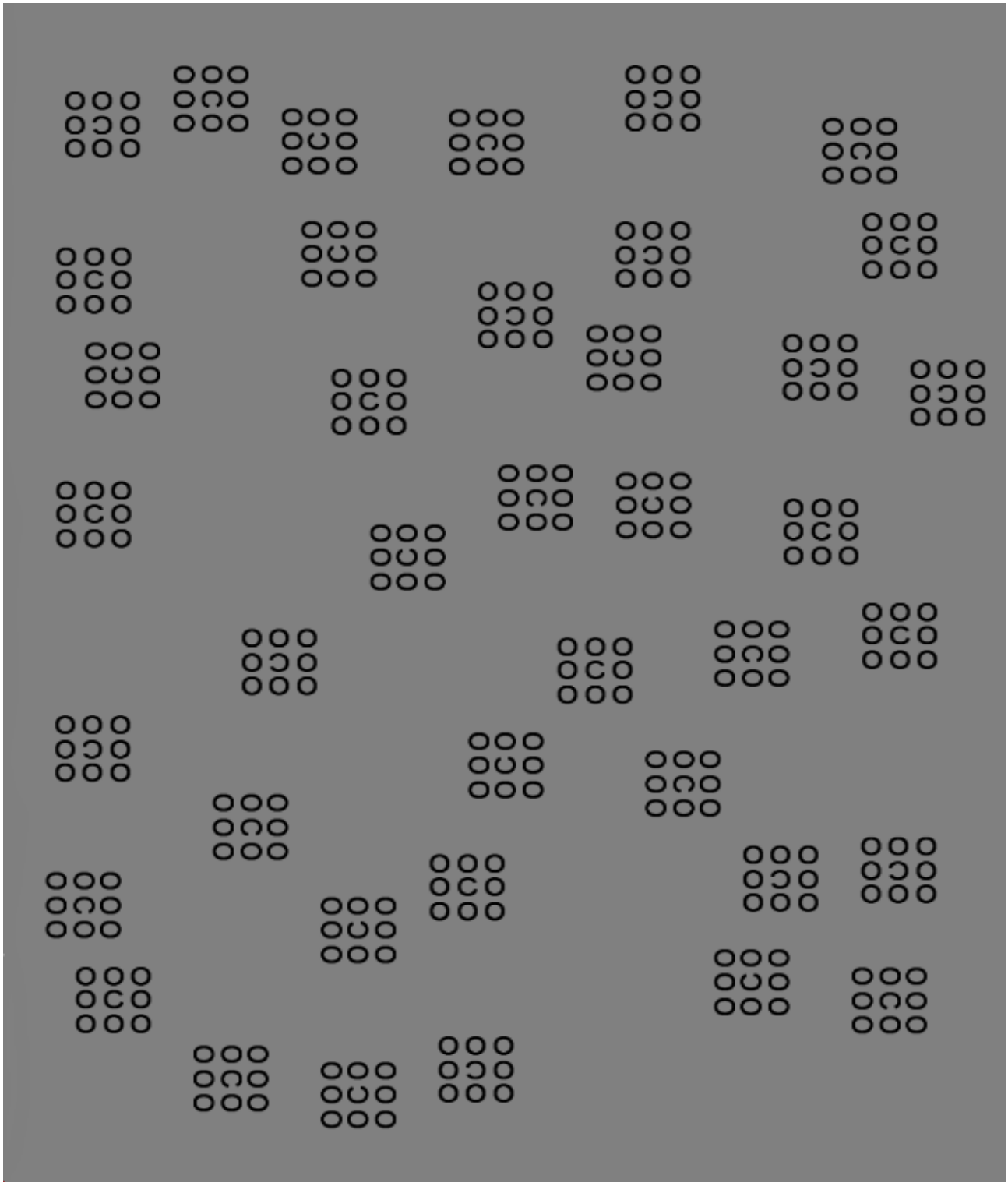

FRPs grand average at Fz
μV
time (s)


FRPs grand average at Cz
μV
time (s)

## 10.2 Appendix B

Appendix B includes a full screenshot for one stimuli presentation for both the original and improved experiment.





## 10.3 Appendix C

Appendix C is the python code for the experiment that was used in this study

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
This experiment was created using PsychoPy2 Experiment Builder (v1.82.01), March 26, 2015, at 10:58
If you publish work using this script please cite the relevant PsychoPy publications
  Peirce, JW (2007) PsychoPy - Psychophysics software in Python. Journal of Neuroscience Methods, 162(1-2), 8-13.
  Peirce, JW (2009) Generating stimuli for neuroscience using PsychoPy.
  Frontiers in Neuroinformatics, 2:10. doi: 10.3389/neuro.11.010.2008
"""
from __future__ import division  # so that 1/3=0.333 instead of 1/3=0
import pickle
from psychopy import visual, core, data, event, logging, sound, gui, tools
from psychopy.constants import *  # things like STARTED, FINISHED
import numpy as np  # whole numpy lib is available, prepend 'np.'
from numpy import sin, cos, tan, log, log10, pi, average, sqrt, std, deg2rad, rad2deg, linspace, asarray
from numpy.random import random, randint, normal, shuffle
import os  # handy system and path functions
import random # used to create the trials at random
import sys
from os.path import expanduser
home = expanduser('~')
sys.path.append('C:/Users/Beheerder/Downloads/python')
from access_midi import *
import FieldTrip
from access_fieldtrip_buffer import *

# connect to midi device for sending hardware markers
midi_sender = init_midi('MIDISPORT 2x2 Port A')
ftc = connect_fieldtrip_buffer('localhost',1971)

# synonym function for conversion from cm to pixels
def toPix (cm):
    return tools.monitorunittools.cm2pix(cm,win.monitor)

def send_event(ftc,c):
    evt = FieldTrip.Event()
    evt.type = 'stimulus'
    evt.value = c
    evt.sample = -1
    ftc.putEvents(evt)

# Ensure that relative paths start from the same directory as this script
_thisDir = os.path.dirname(os.path.abspath(__file__))
os.chdir(_thisDir)

# Store info about the experiment session
expName = u'myexp'  # from the Builder filename that created this script
expInfo = {'participant':'', 'session':'001'}
dlg = gui.DlgFromDict(dictionary=expInfo, title=expName)
if dlg.OK == False: core.quit()  # user pressed cancel
expInfo['date'] = data.getDateStr()  # add a simple timestamp
expInfo['expName'] = expName

# Data file name stem = absolute path + name; later add .psyexp, .csv, .log, etc
filename = _thisDir + os.sep + 'data/%s_%s_%s' %(expInfo['participant'], expName, expInfo['date'])

# An ExperimentHandler isn't essential but helps with data saving
thisExp = data.ExperimentHandler(name=expName, version='',
    extraInfo=expInfo, runtimeInfo=None,
    originPath=None,
    savePickle=True, saveWideText=True,
    dataFileName=filename)
#save a log file for detail verbose info
logFile = logging.LogFile(filename+'.log', level=logging.EXP)
logging.console.setLevel(logging.WARNING)  # this outputs to the screen, not a file

endExpNow = False  # flag for 'escape' or other condition => quit the exp

# Start Code - component code to be run before the window creation

# Setup the Window
win = visual.Window(size=(800, 600), fullscr=True, screen=0, allowGUI=False, allowStencil=False,
    monitor='Lab', color=[0,0,0], colorSpace='rgb',
    blendMode='avg', useFBO=True,
    )
# store frame rate of monitor if we can measure it successfully
expInfo['frameRate']=win.getActualFrameRate()
if expInfo['frameRate']!=None:
    frameDur = 1.0/round(expInfo['frameRate'])
else:
    frameDur = 1.0/60.0 # couldn't get a reliable measure so guess

# Initialize components for Routine "trial" this component generates 30 trials, each
# consisting of  targets
variables=[]
# generate 30 random orientation gradients to be used as target orientations in the trials.
random.seed(45)
randomori=[random.randint(0,3)*90 for x in range (30)]
# creates the list called variables, a list containing 30 trials each consisting of 300 randomly
# places and oriented c's
for a in range (30):
 random.seed(a)
 # generates random orientations for the stimuli.
 placeholder= [random.randint(0,3)*90 for x in range (600)]
 # Removes the orientations which equal the target for this trial,
 # since targets will be assigned their orientation separately
```

```
 orientationlist= [x for x in placeholder if x != randomori[a] ]
 # generaties random locations for stimuli. This needs to be higher
 # than the actual number of stimuli since many will be removed.
 Q = [random.randint(-60,60)/4 for x in range (1000)]
 R = [random.randint(-46,46)/4 for x in range (1000)]
 x=0
 # this loop removes stimuli that are to close to each other.
 # the eyetracker needs about 1cm of space between stimuli.
 while x < len(Q):
  y=x+1
  while y < len (Q):
   if (-1.2<= Q[x]-Q[y] <=1.2) and (-1.2 <= R[x]-R[y] <=1.2):
    Q.pop(y)
    R.pop(y)
    x-=1
    break
   y+=1
  x+=1
 print len(Q)
# trial generates 30 targets and trial 2 generates 270 non targets
 trial=[visual.TextStim(win=win, ori=randomori[a], name='text',
       text=u'c',    font=u'Arial',
       units='pix', pos=[toPix(Q[x]),toPix(R[x])], height=toPix(1), wrapWidth=None,
       color=[-1.000,-1.000,-1.000], colorSpace='rgb', opacity=1,
       depth=-1.0) for x in range (30)]
 trial2=[visual.TextStim(win=win, ori=orientationlist[x-30], name='text',
       text=u'c',    font=u'Arial',
       units='pix', pos=[toPix(Q[x]),toPix(R[x])], height=toPix(1), wrapWidth=None,
       color=[-1.000,-1.000,-1.000], colorSpace='rgb', opacity=1,
       depth=-1.0) for x in range (30, 200)]
 trial.extend(trial2)
 variables.append(trial)

# mouse event, since trials end by mouseclick
mouse = event.Mouse(win=win)
x, y = [None, None]
trialClock= core.Clock()
ISI = core.StaticPeriod(win=win, screenHz=expInfo['frameRate'], name='ISI')

# Initialize components for Routine "Pause". This component tells the participant
# to take a break, lasting as long as they want.
PauseClock = core.Clock()
mouse_2 = event.Mouse(win=win)
x, y = [None, None]
text_3 = visual.TextStim(win=win, ori=0, name='text_3',
    text=u'Pause\n\nPress any mouse button when ready to continue',    font=u'Arial',
    units='cm', pos=[0, 0], height=2, wrapWidth=None,
    color=u'white', colorSpace='rgb', opacity=1,
    depth=-1.0)


# Initialize components for Routine "Target", this components displays to the participant
# what the next trials target orientation is
TargetClock = core.Clock()
text = visual.TextStim(win=win, ori=0, name='text',
    text=u'Your next target is',    font=u'Arial',
    units='cm', pos=[0, 5], height=1, wrapWidth=None,
    color=[-1.000,-1.000,-1.000], colorSpace='rgb', opacity=1,
    depth=0.0)
targetstim=[visual.TextStim(win=win,ori=randomori[x], name='text_2',
    text=u'c',    font=u'Arial',
    units='cm', pos=[0, 0], height=5, wrapWidth=None,
    color=[-1.000,-1.000,-1.000], colorSpace='rgb', opacity=1,
    depth=-1.0) for x in range (30)]

# the random order is printed in order to trace back the order trials were presented in
# N.B  change program to save this order in a file instead of printing
print randomori

# Create some handy timers
globalClock = core.Clock()  # to track the time since experiment started
routineTimer = core.CountdownTimer()  # to track time remaining of each (non-slip) routine

# create Expnr a shuffled list of nrs 0 to 30 in order to be able to call trials in a random
# order. Changing the seed allows you to generate a new random order for each participant
random.seed(int(expInfo['session']))
x= [[i] for i in range (30)]
random.shuffle(x)
Expnr=[x[i][0] for i in range (30)]


for z in range (30):
    #------Prepare to start Routine "Target"-------
    t = 0
    TargetClock.reset()  # clock
    frameN = -1
    routineTimer.add(5.000000)
    # update component parameters for each repeat
    # keep track of which components have finished
    TargetComponents = []
    TargetComponents.append(text)
    TargetComponents.append(targetstim[Expnr[z]])
    for thisComponent in TargetComponents:
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED

    #-------Start Routine "Target"-------
    continueRoutine = True
    while continueRoutine and routineTimer.getTime() > 0:
        # get current time
        t = TargetClock.getTime()
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
```

```
        # update/draw components on each frame

        # *text* updates
        if t >= 0.0 and text.status == NOT_STARTED:
            # keep track of start time/frame for later
            text.tStart = t  # underestimates by a little under one frame
            text.frameNStart = frameN  # exact frame index
            text.setAutoDraw(True)
        if text.status == STARTED and t >= (5.0-win.monitorFramePeriod*0.75): #most of one frame period left
            text.setAutoDraw(False)

        # *text_2* updates
        if t >= 0.0 and targetstim[z].status == NOT_STARTED:
            # keep track of start time/frame for later
            targetstim[Expnr[z]].tStart = t  # underestimates by a little under one frame
            targetstim[Expnr[z]].frameNStart = frameN  # exact frame index
            targetstim[Expnr[z]].setAutoDraw(True)
        if targetstim[Expnr[z]].status == STARTED and t >= (0.0 + (5.0-win.monitorFramePeriod*0.75)): #most of one frame period left
            targetstim[Expnr[z]].setAutoDraw(False)

        # check if all components have finished
        if not continueRoutine:  # a component has requested a forced-end of Routine
            break
        continueRoutine = False  # will revert to True if at least one component still running
        for thisComponent in TargetComponents:
            if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
                continueRoutine = True
                break  # at least one component has not yet finished

        # check for quit (the Esc key)
        if endExpNow or event.getKeys(keyList=["escape"]):
            core.quit()

        # refresh the screen
        if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
            win.flip()

#-------Ending Routine "Target"-------
    for thisComponent in TargetComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)

        #------Prepare to start Routine "trial"-------
    t = 0
    trialClock.reset()  # clock
    frameN = -1
    # update component parameters for each repeat
    # setup some python lists for storing info about the mouse
    # keep track of which components have finished
    trialComponents = []
    trialComponents.append(ISI)
    for x in range (200):
     trialComponents.append(variables[Expnr[z]][x])
    trialComponents.append(mouse)
    for thisComponent in trialComponents:
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED

    #-------Start Routine "trial"-------
    # send midi marker
    value = z

     send_event(ftc,marker)
     biosemimarker = send_midi_marker(midi_sender,'stimulus',z+1,'biosemi_active2')

     continueRoutine = True
    while continueRoutine:
        # get current time
        t = trialClock.getTime()
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame

        # *text* updates
        for x in range (200):
         if t >= 1 and variables[Expnr[z]][x].status == NOT_STARTED:
            # keep track of start time/frame for later
            variables[Expnr[z]][x].tStart = t  # underestimates by a little under one frame
            variables[Expnr[z]][x].frameNStart = frameN  # exact frame index
            variables[Expnr[z]][x].setAutoDraw(True)
         if variables[Expnr[z]][x].status == STARTED and (mouse==PRESSED):
            variables[Expnr[z]][x].setAutoDraw(False)
        # *mouse* updates
        if t >= 0.0 and mouse.status == NOT_STARTED:
            # keep track of start time/frame for later
            mouse.tStart = t  # underestimates by a little under one frame
            mouse.frameNStart = frameN  # exact frame index
            mouse.status = STARTED
            event.mouseButtons = [0, 0, 0]  # reset mouse buttons to be 'up'
        if mouse.status == STARTED:  # only update if started and not stopped!
            buttons = mouse.getPressed()
            if sum(buttons) > 0:  # ie if any button is pressed
                # abort routine on response
                continueRoutine = False
        # *ISI* period
        if t >= 0.0 and ISI.status == NOT_STARTED:
            # keep track of start time/frame for later
            ISI.tStart = t  # underestimates by a little under one frame
            ISI.frameNStart = frameN  # exact frame index
            ISI.start(1)
        elif ISI.status == STARTED: #one frame should pass before updating params and completing
            ISI.complete() #finish the static period
```

```
    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in trialComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # check for quit (the Esc key)
    if endExpNow or event.getKeys(keyList=["escape"]):
        core.quit()

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

#-------Ending Routine "trial"-------
for thisComponent in trialComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# send midi marker
value = z

 send_event(ftc,marker)
 biosemimarker = send_midi_marker(midi_sender,'stimulus',z+1,'biosemi_active2')

# store data for thisExp (ExperimentHandler)
 x, y = mouse.getPos()
buttons = mouse.getPressed()
thisExp.addData('mouse.x', x)
thisExp.addData('mouse.y', y)
thisExp.addData('mouse.leftButton', buttons[0])
thisExp.addData('mouse.midButton', buttons[1])
thisExp.addData('mouse.rightButton', buttons[2])
thisExp.addData('Trialnumber',Expnr[z])
thisExp.addData('Targetlocations',[[variables[Expnr[z]][xs].pos[0]
+400,abs(variables[Expnr[z]][xs].pos[1]-300)] for xs in range (30)])
thisExp.nextEntry()
# the Routine "trial" was not non-slip safe, so reset the non-slip timer
routineTimer.reset()


#------Prepare to start Routine "Pause"-------
t = 0
PauseClock.reset()  # clock
frameN = -1
# update component parameters for each repeat
# setup some python lists for storing info about the mouse_2
# keep track of which components have finished
PauseComponents = []
PauseComponents.append(mouse_2)
PauseComponents.append(text_3)
for thisComponent in PauseComponents:
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED

#-------Start Routine "Pause"-------
continueRoutine = True
while continueRoutine:
    # get current time
    t = PauseClock.getTime()
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame
    # *mouse_2* updates
    if t >= 0.0 and mouse_2.status == NOT_STARTED:
        # keep track of start time/frame for later
        mouse_2.tStart = t  # underestimates by a little under one frame
        mouse_2.frameNStart = frameN  # exact frame index
        mouse_2.status = STARTED
        event.mouseButtons = [0, 0, 0]  # reset mouse buttons to be 'up'
    if mouse_2.status == STARTED:  # only update if started and not stopped!
        buttons = mouse_2.getPressed()
        if sum(buttons) > 0:  # ie if any button is pressed
            # abort routine on response
            continueRoutine = False

    # *text_3* updates
    if t >= 0.0 and text_3.status == NOT_STARTED:
        # keep track of start time/frame for later
        text_3.tStart = t  # underestimates by a little under one frame
        text_3.frameNStart = frameN  # exact frame index
        text_3.setAutoDraw(True)
    if text_3.status == STARTED and (mouse==PRESSED):
        text_3.setAutoDraw(False)

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in PauseComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # check for quit (the Esc key)
    if endExpNow or event.getKeys(keyList=["escape"]):
        core.quit()

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()
```

```
    #-------Ending Routine "Pause"-------
    for thisComponent in PauseComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)
    # store data for thisExp (ExperimentHandler)
    x, y = mouse_2.getPos()
    buttons = mouse_2.getPressed()
    thisExp.addData('mouse_2.x', x)
    thisExp.addData('mouse_2.y', y)
    thisExp.addData('mouse_2.leftButton', buttons[0])
    thisExp.addData('mouse_2.midButton', buttons[1])
    thisExp.addData('mouse_2.rightButton', buttons[2])
    thisExp.nextEntry()
    # the Routine "Pause" was not non-slip safe, so reset the non-slip timer
    routineTimer.reset()
win.close()
core.quit()
```

## 10.4 Appendix D

Appendix D includes the full code for the improved PsychoPy experiment.

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
This experiment was created using PsychoPy2 Experiment Builder (v1.82.01), March 26, 2015, at 10:58
If you publish work using this script please cite the relevant PsychoPy publications
  Peirce, JW (2007) PsychoPy - Psychophysics software in Python. Journal of Neuroscience Methods, 162(1-2), 8-13.
  Peirce, JW (2009) Generating stimuli for neuroscience using PsychoPy.
Frontiers in Neuroinformatics, 2:10. doi: 10.3389/neuro.11.010.2008
"""
from __future__ import division  # so that 1/3=0.333 instead of 1/3=0
import pickle
from psychopy import visual, core, data, event, logging, sound, gui, tools
from psychopy.constants import *  # things like STARTED, FINISHED
import numpy as np  # whole numpy lib is available, prepend 'np.'
from numpy import sin, cos, tan, log, log10, pi, average, sqrt, std, deg2rad, rad2deg, linspace, asarray
from numpy.random import random, randint, normal, shuffle
import os  # handy system and path functions
import random # used to create the trials at random
import sys
from os.path import expanduser
home = expanduser('~')
sys.path.append('/Network/Servers/mmmxserver.nici.ru.nl/Volumes/Xserver_RAID/Users/danny/
bci_code/toolboxes/brainstream/modules/python')
from access_midi import *
import FieldTrip
from access_fieldtrip_buffer import *

# connect to midi device for sending hardware markers
midi_sender = init_midi('828mk2 MIDI Port')
ftc = connect_fieldtrip_buffer('localhost',1971)

# synonym function for conversion from cm to pixels
def toPix (cm):
    return tools.monitorunittools.cm2pix(cm,win.monitor)

def send_event(ftc,c):
    evt = FieldTrip.Event()
    evt.type = 'stimulus'
    evt.value = c
    evt.sample = -1
    ftc.putEvents(evt)

# Ensure that relative paths start from the same directory as this script
_thisDir = os.path.dirname(os.path.abspath(__file__))
os.chdir(_thisDir)

# Store info about the experiment session
expName = u'myexp'  # from the Builder filename that created this script
expInfo = {'participant':'', 'session':'001'}
dlg = gui.DlgFromDict(dictionary=expInfo, title=expName)
if dlg.OK == False: core.quit()  # user pressed cancel
expInfo['date'] = data.getDateStr()  # add a simple timestamp
expInfo['expName'] = expName

# Data file name stem = absolute path + name; later add .psyexp, .csv, .log, etc
filename = _thisDir + os.sep + 'data/%s_%s_%s' %(expInfo['participant'], expName, expInfo['date'])

# An ExperimentHandler isn't essential but helps with data saving
thisExp = data.ExperimentHandler(name=expName, version='',
    extraInfo=expInfo, runtimeInfo=None,
    originPath=None,
    savePickle=True, saveWideText=True,
    dataFileName=filename)
#save a log file for detail verbose info
logFile = logging.LogFile(filename+'.log', level=logging.EXP)
logging.console.setLevel(logging.WARNING)  # this outputs to the screen, not a file

endExpNow = False  # flag for 'escape' or other condition => quit the exp

# Start Code - component code to be run before the window creation

# Setup the Window
win = visual.Window(size=(800, 600), fullscr=False, screen=1, allowGUI=False, allowStencil=False,
    monitor='Lab', color=[0,0,0], colorSpace='rgb',
    blendMode='avg', useFBO=True,
    )
# store frame rate of monitor if we can measure it successfully
expInfo['frameRate']=win.getActualFrameRate()
if expInfo['frameRate']!=None:
    frameDur = 1.0/round(expInfo['frameRate'])
else:
    frameDur = 1.0/60.0 # couldn't get a reliable measure so guess

# Initialize components for Routine "trial" this component generates 30 trials, each
# consisting of  targets
variables=[]
# generate 30 random orientation gradients to be used as target orientations in the trials.
random.seed(45)
randomori=[random.randint(0,3)*90 for x in range (30)]
# creates the list called variables, a list containing 30 trials each consisting of 300 randomly
# places and oriented c's
for a in range (30):
 random.seed(a)
 # generates random orientations for the stimuli.
 placeholder= [random.randint(0,3)*90 for x in range (600)]
 # Removes the orientations which equal the target for this trial,
 # since targets will be assigned their orientation separately
```

```
 orientationlist= [x for x in placeholder if x != randomori[a] ]
 # generaties random locations for stimuli. This needs to be higher
 # than the actual number of stimuli since many will be removed.
 Q = [random.randint(-59,59)/4 for x in range (1000)]
 R = [random.randint(-45,45)/4 for x in range (1000)]
 x=0
 # this loop removes stimuli that are to close to each other.
 # the eyetracker needs about 1cm of space between stimuli.
 while x < len(Q):
  y=x+1
  while y < len (Q):
   if (-2.6<= Q[x]-Q[y] <=2.6) and (-2.6 <= R[x]-R[y] <=2.6):
    Q.pop(y)
    R.pop(y)
    x-=1
    break
   y+=1
  x+=1
# trial generates targets and trial 2 generates non targets
 trial=[visual.ImageStim(win=win, name='image',units='pix',
     image=u'C:\\Users\\Beheerder\\School\\KI\\Scriptie\\Psychopy\\stim.png', mask=None,
     ori=randomori[a], pos=[toPix(Q[x]),toPix(R[x])], size=[toPix(2), toPix(2)],
     color=[1,1,1], colorSpace='rgb', opacity=1,
     flipHoriz=False, flipVert=False,
     texRes=128, interpolate=True, depth=-1.0) for x in range (10)]
 trial2=[visual.ImageStim(win=win, name='image',units='pix',
     image=u'C:\\Users\\Beheerder\\School\\KI\\Scriptie\\Psychopy\\stim.png', mask=None,
     ori=orientationlist[x-30], pos=[toPix(Q[x]),toPix(R[x])], size=[toPix(2), toPix(2)],
     color=[1,1,1], colorSpace='rgb', opacity=1,
     flipHoriz=False, flipVert=False,
     texRes=128, interpolate=True, depth=-1.0) for x in range (10, 40)]
 trial.extend(trial2)
 variables.append(trial)

# mouse event, since trials end by mouseclick
mouse = event.Mouse(win=win)
x, y = [None, None]
trialClock= core.Clock()
ISI = core.StaticPeriod(win=win, screenHz=expInfo['frameRate'], name='ISI')

# Initialize components for Routine "Pause". This component tells the participant
# to take a break, lasting as long as they want.
PauseClock = core.Clock()
mouse_2 = event.Mouse(win=win)
x, y = [None, None]
text_3 = visual.TextStim(win=win, ori=0, name='text_3',
     text=u'Pause\n\nPress any mouse button when ready to continue',    font=u'Arial',
     units='cm', pos=[0, 0], height=2, wrapWidth=None,
     color=u'white', colorSpace='rgb', opacity=1,
     depth=-1.0)


# Initialize components for Routine "Target", this components displays to the participant
# what the next trials target orientation is
TargetClock = core.Clock()
text = visual.TextStim(win=win, ori=0, name='text',
     text=u'Your next target is',    font=u'Arial',
     units='cm', pos=[0, 5], height=1, wrapWidth=None,
     color=[-1.000,-1.000,-1.000], colorSpace='rgb', opacity=1,
     depth=0.0)
targetstim=[visual.TextStim(win=win,ori=randomori[x], name='text_2',
     text=u'c',    font=u'Arial',
     units='cm', pos=[0, 0], height=5, wrapWidth=None,
     color=[-1.000,-1.000,-1.000], colorSpace='rgb', opacity=1,
     depth=-1.0) for x in range (30)]

# the random order is printed in order to trace back the order trials were presented in
# N.B  change program to save this order in a file instead of printing


# Create some handy timers
globalClock = core.Clock()  # to track the time since experiment started
routineTimer = core.CountdownTimer()  # to track time remaining of each (non-slip) routine

# create Expnr a shuffled list of nrs 0 to 30 in order to be able to call trials in a random
# order. Changing the seed allows you to generate a new random order for each participant
random.seed(int(expInfo['session']))
x= [[i] for i in range (30)]
random.shuffle(x)
Expnr=[x[i][0] for i in range (30)]

marker = 0
for z in range (30):
    marker =marker+1
    #-------Prepare to start Routine "Target"--------
    t = 0
    TargetClock.reset()  # clock
    frameN = -1
    routineTimer.add(1.000000)
    # update component parameters for each repeat
    # keep track of which components have finished
    TargetComponents = []
    TargetComponents.append(text)
    TargetComponents.append(targetstim[Expnr[z]])
    for thisComponent in TargetComponents:
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED

    #-------Start Routine "Target"--------
    continueRoutine = True
    while continueRoutine and routineTimer.getTime() > 0:
        # get current time
```

```
        t = TargetClock.getTime()
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame

        # *text* updates
        if t >= 0.0 and text.status == NOT_STARTED:
            # keep track of start time/frame for later
            text.tStart = t  # underestimates by a little under one frame
            text.frameNStart = frameN  # exact frame index
            text.setAutoDraw(True)
        if text.status == STARTED and t >= (5.0-win.monitorFramePeriod*0.75): #most of one frame period left
            text.setAutoDraw(False)

        # *text_2* updates
        if t >= 0.0 and targetstim[Expnr[z]].status == NOT_STARTED:
            # keep track of start time/frame for later
            targetstim[Expnr[z]].tStart = t  # underestimates by a little under one frame
            targetstim[Expnr[z]].frameNStart = frameN  # exact frame index
            targetstim[Expnr[z]].setAutoDraw(True)
        if targetstim[Expnr[z]].status == STARTED and t >= (0.0 + (5.0-win.monitorFramePeriod*0.75)): #most of one frame period left
            targetstim[Expnr[z]].setAutoDraw(False)

        # check if all components have finished
        if not continueRoutine:  # a component has requested a forced-end of Routine
            break
        continueRoutine = False  # will revert to True if at least one component still running
        for thisComponent in TargetComponents:
            if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
                continueRoutine = True
                break  # at least one component has not yet finished

        # check for quit (the Esc key)
        if endExpNow or event.getKeys(keyList=["escape"]):
            core.quit()

        # refresh the screen
        if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
            win.flip()

#-------Ending Routine "Target"-------
    for thisComponent in TargetComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)

        #------Prepare to start Routine "trial"-------
    t = 0
    trialClock.reset()  # clock
    frameN = -1
    # update component parameters for each repeat
    # setup some python lists for storing info about the mouse
    # keep track of which components have finished
    trialComponents = []
    trialComponents.append(ISI)
    for x in range (40):
     trialComponents.append(variables[Expnr[z]][x])
    trialComponents.append(mouse)
    for thisComponent in trialComponents:
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED

    #-------Start Routine "trial"-------
    # send midi marker
    biosemimarker = send_midi_marker(midi_sender,'stimulus',marker,'biosemi_active2')
    send_event(ftc,marker)
    print marker
    continueRoutine = True
    while continueRoutine:
        # get current time
        t = trialClock.getTime()
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame

        # *text* updates
        for x in range (40):
         if t >= 1 and variables[Expnr[z]][x].status == NOT_STARTED:
            # keep track of start time/frame for later
            variables[Expnr[z]][x].tStart = t  # underestimates by a little under one frame
            variables[Expnr[z]][x].frameNStart = frameN  # exact frame index
            variables[Expnr[z]][x].setAutoDraw(True)
         if variables[Expnr[z]][x].status == STARTED and (mouse==PRESSED):
            variables[Expnr[z]][x].setAutoDraw(False)
        # *mouse* updates
        if t >= 0.0 and mouse.status == NOT_STARTED:
            # keep track of start time/frame for later
            mouse.tStart = t  # underestimates by a little under one frame
            mouse.frameNStart = frameN  # exact frame index
            mouse.status = STARTED
            event.mouseButtons = [0, 0, 0]  # reset mouse buttons to be 'up'
        if mouse.status == STARTED:  # only update if started and not stopped!
            buttons = mouse.getPressed()
            if sum(buttons) > 0:  # ie if any button is pressed
                # abort routine on response
                continueRoutine = False
        # *ISI* period
        if t >= 0.0 and ISI.status == NOT_STARTED:
            # keep track of start time/frame for later
            ISI.tStart = t  # underestimates by a little under one frame
            ISI.frameNStart = frameN  # exact frame index
            ISI.start(1)
        elif ISI.status == STARTED: #one frame should pass before updating params and completing
            ISI.complete() #finish the static period
```

```
    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in trialComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # check for quit (the Esc key)
    if endExpNow or event.getKeys(keyList=["escape"]):
        core.quit()

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

#-------Ending Routine "trial"-------
for thisComponent in trialComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# send midi marker
biosemimarker = send_midi_marker(midi_sender,'stimulus',marker,'biosemi_active2')
send_event(ftc,marker)
# store data for thisExp (ExperimentHandler)
x, y = mouse.getPos()
buttons = mouse.getPressed()
thisExp.addData('mouse.x', x)
thisExp.addData('mouse.y', y)
thisExp.addData('mouse.leftButton', buttons[0])
thisExp.addData('mouse.midButton', buttons[1])
thisExp.addData('mouse.rightButton', buttons[2])
thisExp.addData('Trialnumber',Expnr[z])
thisExp.addData('Targetlocations',[[variables[Expnr[z]][xs].pos[0]+400,
abs(variables[Expnr[z]][xs].pos[1]-300)] for xs in range (10)])
thisExp.addData('Distractorlocations',[[variables[Expnr[z]][xs].pos[0]+400,
abs(variables[Expnr[z]][xs].pos[1]-300)] for xs in range (10,40)])
thisExp.nextEntry()
# the Routine "trial" was not non-slip safe, so reset the non-slip timer
routineTimer.reset()

marker= marker+1
#------Prepare to start Routine "trial2"-------
t = 0
trialClock.reset()  # clock
frameN = -1
# update component parameters for each repeat
# setup some python lists for storing info about the mouse
# keep track of which components have finished
trialComponents = []
trialComponents.append(ISI)
variables[Expnr[z]][10].ori = variables[Expnr[z]][9].ori
for x in range (40):
 trialComponents.append(variables[Expnr[z]][x])
trialComponents.append(mouse)
for thisComponent in trialComponents:
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED

#-------Start Routine "trial2"-------
# send midi marker
biosemimarker = send_midi_marker(midi_sender,'stimulus',marker,'biosemi_active2')
send_event(ftc,marker)
print marker
continueRoutine = True
while continueRoutine:
    # get current time
    t = trialClock.getTime()
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame

    # *text* updates
    for x in range (40):
     if t >= 1 and variables[Expnr[z]][x].status == NOT_STARTED:
        # keep track of start time/frame for later
        variables[Expnr[z]][x].tStart = t  # underestimates by a little under one frame
        variables[Expnr[z]][x].frameNStart = frameN  # exact frame index
        variables[Expnr[z]][x].setAutoDraw(True)
     if variables[Expnr[z]][x].status == STARTED and (mouse==PRESSED):
        variables[Expnr[z]][x].setAutoDraw(False)
    # *mouse* updates
    if t >= 0.0 and mouse.status == NOT_STARTED:
        # keep track of start time/frame for later
        mouse.tStart = t  # underestimates by a little under one frame
        mouse.frameNStart = frameN  # exact frame index
        mouse.status = STARTED
        event.mouseButtons = [0, 0, 0]  # reset mouse buttons to be 'up'
    if mouse.status == STARTED:  # only update if started and not stopped!
        buttons = mouse.getPressed()
        if sum(buttons) > 0:  # ie if any button is pressed
            # abort routine on response
            continueRoutine = False
    # *ISI* period
    if t >= 0.0 and ISI.status == NOT_STARTED:
        # keep track of start time/frame for later
        ISI.tStart = t  # underestimates by a little under one frame
        ISI.frameNStart = frameN  # exact frame index
        ISI.start(1)
    elif ISI.status == STARTED: #one frame should pass before updating params and completing
        ISI.complete() #finish the static period

    # check if all components have finished
```

```
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in trialComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # check for quit (the Esc key)
    if endExpNow or event.getKeys(keyList=["escape"]):
        core.quit()

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

#-------Ending Routine "trial2"-------
for thisComponent in trialComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# send midi marker
biosemimarker = send_midi_marker(midi_sender,'stimulus',marker,'biosemi_active2')
send_event(ftc,marker)
# store data for thisExp (ExperimentHandler)
x, y = mouse.getPos()
buttons = mouse.getPressed()
thisExp.addData('mouse.x', x)
thisExp.addData('mouse.y', y)
thisExp.addData('mouse.leftButton', buttons[0])
thisExp.addData('mouse.midButton', buttons[1])
thisExp.addData('mouse.rightButton', buttons[2])
thisExp.addData('Trialnumber',Expnr[z])
thisExp.nextEntry()
# the Routine "trial" was not non-slip safe, so reset the non-slip timer
routineTimer.reset()    #------Prepare to start Routine "Pause"-------
t = 0
PauseClock.reset()  # clock
frameN = -1
# update component parameters for each repeat
# setup some python lists for storing info about the mouse_2
# keep track of which components have finished
PauseComponents = []
PauseComponents.append(mouse_2)
PauseComponents.append(text_3)
for thisComponent in PauseComponents:
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED

#-------Start Routine "Pause"-------
continueRoutine = True
while continueRoutine:
    # get current time
    t = PauseClock.getTime()
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame
    # *mouse_2* updates
    if t >= 0.0 and mouse_2.status == NOT_STARTED:
        # keep track of start time/frame for later
        mouse_2.tStart = t  # underestimates by a little under one frame
        mouse_2.frameNStart = frameN  # exact frame index
        mouse_2.status = STARTED
        event.mouseButtons = [0, 0, 0]  # reset mouse buttons to be 'up'
    if mouse_2.status == STARTED:  # only update if started and not stopped!
        buttons = mouse_2.getPressed()
        if sum(buttons) > 0:  # ie if any button is pressed
            # abort routine on response
            continueRoutine = False

    # *text_3* updates
    if t >= 0.0 and text_3.status == NOT_STARTED:
        # keep track of start time/frame for later
        text_3.tStart = t  # underestimates by a little under one frame
        text_3.frameNStart = frameN  # exact frame index
        text_3.setAutoDraw(True)
    if text_3.status == STARTED and (mouse==PRESSED):
        text_3.setAutoDraw(False)

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in PauseComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # check for quit (the Esc key)
    if endExpNow or event.getKeys(keyList=["escape"]):
        core.quit()

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

#-------Ending Routine "Pause"-------
for thisComponent in PauseComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# store data for thisExp (ExperimentHandler)
x, y = mouse_2.getPos()
buttons = mouse_2.getPressed()
thisExp.addData('mouse_2.x', x)
```

```
    thisExp.addData('mouse_2.y', y)
    thisExp.addData('mouse_2.leftButton', buttons[0])
    thisExp.addData('mouse_2.midButton', buttons[1])
    thisExp.addData('mouse_2.rightButton', buttons[2])
    thisExp.nextEntry()
    # the Routine "Pause" was not non-slip safe, so reset the non-slip timer
    routineTimer.reset()
win.close()
core.quit()
```