



RADBOUD UNIVERSITY NIJMEGEN

BACHELOR THESIS OF ARTIFICIAL INTELLIGENCE

**SSVEP-based BCI control for
navigating a robot**

Author:
G.G.J. Jacobs

Supervisors:
dr. J. Farquhar
dr. L.G. Vuurpijl

December 2013

Contents

1	Introduction	4
2	Brain-Computer Interfaces	6
2.1	Modality	6
2.2	Pre-processing	7
2.3	Feature extraction	7
2.4	Prediction	7
2.5	Output	8
3	Steady-State Visually Evoked Potentials	9
4	Experiment: robot navigation	11
4.1	Introduction	11
4.2	Requirements & details	11
4.3	Tasks	12
4.4	Experiment	13
4.4.1	Pairing the robot	13
4.4.2	Initialization	14
4.4.3	Calibration	14
4.4.4	Training the classifier	15
4.4.5	Signal pre-processing	15
4.4.5.1	De-trending	15
4.4.5.2	Bad channel removal	15
4.4.5.3	Re-referencing	16
4.4.5.4	Bad trial removal	16
4.4.5.5	Welch's method	16
4.4.5.6	Frequency range selection	16
4.4.6	Classification	16
4.4.7	LEGO NXT robot	17
4.5	Results	17
5	Experiment: stimulation at a distance	19
5.1	Introduction	19
5.2	Requirements & details	19
5.3	Tasks	19
5.4	Experiment	20
5.5	Results	20
6	Conclusion - Discussion	25
6.1	Improvements and future work	25

7	Acknowledgements	27
8	References	28
9	Appendix	31
9.1	Robot files	31
9.2	Buffer_BCI adaptations	34
9.3	Data processing and plotting files	44

Abstract

Controlling devices without psychical contact is an important aspect of paralyzed people, operators who need to handle multiple controls and those missing limbs caused by an accident. Speech recognition can work for some, but is not efficient in noisy areas. BCI systems using imagined movement can overcome this, but it is tiring to keep a constant imagined state that is mentally clear enough for the system to detect. Steady-state visually evoked potentials (SSVEPs) are known to induce a clear signal for a BCI system to trigger external devices or mechanics. This way, someone can control a mobile device by merely looking at a particular target on the device.

This research will answer the question of how BCI performance behaves over larger distances (>100 cm). More specific, we designed an experiment that would test how both the user-stimulus distance and orientation of the stimuli would affect SSVEP BCI performance. For this we used a Lego robot application to navigate a multiple-turn track. Performance rates during navigation varied between 59.0%-70.4% with large differences in completion times among participants. As our question remained unanswered, we conducted a second experiment with a focus on measuring the relation between user-stimulus distance and SSVEP BCI performance. Results show a relatively stable signal for non-target stimuli, while the target stimulus showed a decrease in frequency amplitude over distance. The majority of the obtained results shows a significant correlation between the distance, which is related to the visual angle, and the amplitude of the stimulated frequency.

Keywords: BCI, brain-computer interface, SSVEP, steady-state visually evoked potential, LED, distance, visual angle, LEGO robot.

1 Introduction

Brain-computer interfaces have shown being of great importance for those who cannot communicate physically. Those who only have their potential brain at hand can hugely benefit as BCIs establish an additional way of interacting through a direct connection between the human brain and a computer.

Typically, patients diagnosed with amyotrophic lateral sclerosis (ALS), a motor neuron disease causing a decreased motor control which eventually leads to a complete locked-in state, have been asked to participate in BCI experiments. Results indicate that BCI performance for ALS patients is maintained throughout the different stages of paralysis (Silvoni et al., 2009, 2013).

Besides ALS patients, research shows that people suffering from a spinal cord injury are able to successfully control a BCI system (Leeb et al., 2007; Conradi et al., 2009). The severity of lack of muscle control for these participants ranges from decreased leg control to a total loss of function of all four limbs (tetraplegia). This means a BCI can fulfill the specific needs of each movement-disabled person, i.e. using a robotic arm for grasping or controlling a wheelchair for navigation.

Several examples show successful use of steady-state visually evoked potentials in BCI systems (Ortner et al., 2011; G. R. Muller-Putz & Pfurtscheller, 2008; Pfurtscheller et al., 2010). SSVEP-based systems do not require extensive training sessions, but only require participants to focus attention on the flickering lights. A common factor in most SSVEP BCI studies is that the stimulus is at a fixed distance near to the subject, generally less than 100 cm (Wu et al., 2008; Resalat & Setarehdan, 2012; Müller et al., 1998; Kelly et al., 2005).

In this study we wish to find an answer to the question if it is feasible to work with a SSVEP BCI-controlled application that extends beyond a distance of 100 cm. Likely, there will be a relationship between the distance, or visual angle of stimulation, and the corresponding amplitude of a stimulus.

Besides, at larger distances the relative space between neighbouring stimuli decreases which could result in overlapping, thus competing, stimulation on the retina (especially the fovea).

As indicated above, we would like to investigate the influence of distant stimulation on the performance of a SSVEP BCI. In particular, we will try to answer the following questions:

- How does the distance between the user and the robot affect performance?
- How does the orientation of the stimuli (LEDs) affect performance?

This work consists of several chapters each describing its own subject. Chapter 2 will give an overview of what a brain-computer interface is, what steps the system takes from stimulus to output and issues that BCIs face during operation. Then, Chapter 3 will describe the nature of steady-evoked visually evoked potentials, the type of stimulus being used in the experiments, and the main advantage of SSVEP-based systems. Next, Chapter 4 will explain the purpose of our first experiment along with its details and used hardware to find out the influence of distance and orientation of stimuli on classification performance. This experiment included the navigation of a LEGO robot through a multiple-turn track. Following the navigation experiment, in Chapter 5 a second experiment

was done that mainly focussed on the relationship between the distance and both the frequency power and performance rate of the SSVEP BCI. Finally, in Chapter 6 the conclusions from both experiments are shared with the reader. To conclude this writing several suggestions and improvements to our work are given to continue future research.

2 Brain-Computer Interfaces

This chapter will give an introduction to brain-computer interfaces (BCI). As BCIs require several steps to be taken for successful operation, these are described one by one to explain what the role of each in the BCI cycle is.

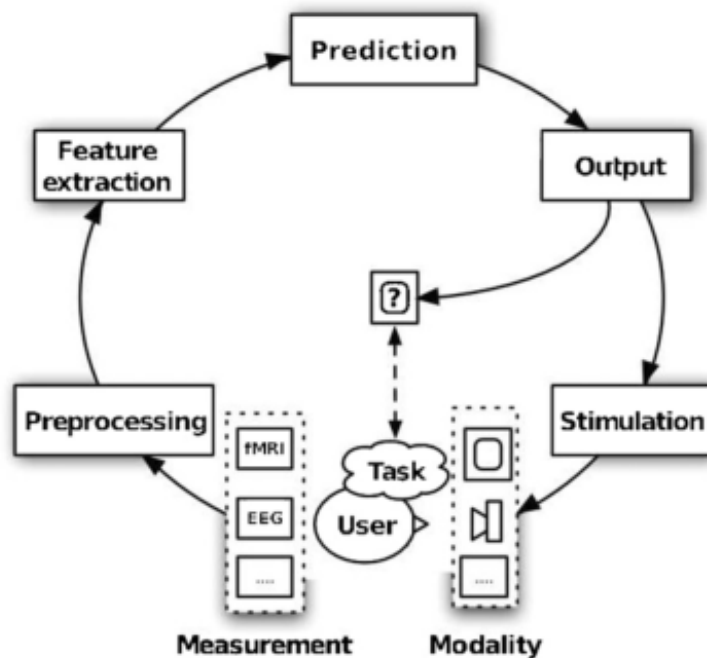


Figure 1: The brain-computer interface cycle (van Gerven et al., 2009).

Figure 1 shows an overview of the classic BCI cycle as used in van Gerven et al. (2009). Each step in the process has its own function as it closes the complete cycle. For every intention the user has, the system will follow each step of the cycle to be able to output information on-screen or to actuate a movement of a physical device.

2.1 Modality

As the BCI cycle shows, several modalities can be used as stimulation to extract evoked or event-related data from the user. Among these modalities are visual, auditory and somatosensory (tactile) stimulation. In our research, steady-state evoked potentials are used for brain-computer communication. These brain potentials are evoked when one is offered a type of stimulation with a periodic pattern. This category of stimulation can be either auditory (ASSEP), somatosensory (SSSEP) or visually (SSVEP) presented. For examples of different steady-state evoked potential modalities used in BCI systems, see Farquhar et al. (2008); Lins & Picton (1995); Patel & Balaban (2004); G. Muller-Putz et al. (2006); Lalor et al. (2005); Müller et al. (1998).

Besides stimulus-based potentials, another way to control BCIs is through imagined movement. With an IM-based BCI system, the user imagines performing a movement which in turn accomplishes a signal in the brain that can be detected by the system (for an example see Blankertz et al. (2010)).

During our experiments we offered brain-computer communication of participants by use of the steady-state visually evoked potentials. An overview and neuronal background of this type of evoked potentials will be given in chapter 3.

2.2 Pre-processing

The brain signals obtained by the measurement device, have to be pre-processed. Raw brain data contains a lot of noise caused by eye movements, muscle contractions and external sources of signal noise such as 50-60 Hz line noise . These contaminations of the relevant signals have to be cleaned by special filters. A spectral filter is used to remove unwanted sources of noise such as slow drift and line noise. For aiming at the signal at a particular location in the brain, a spatial filter is used to reject certain electrodes irrelevant for the given position. Besides spectral and spatial filters, artifact detection is used to detect and reject external signal noise such as eye or muscle artifacts.

2.3 Feature extraction

The pre-processed signals can now be used for extracting any features of interest. Features such as the amplitude of a particular frequency is used for predicting outcomes by the system. Temporal features focus on the signal over time, while spectral features characterize the signal in various frequency bands. A time-frequency representation combines both temporal and spectral features to show how signals behave in amplitude over time.

2.4 Prediction

Since the main purpose of the BCI system is to control something by pure brain signals, whether this is a prosthetic arm or cursor on a computer screen, the most important part of the cycle might be the prediction.

Deriving which output relates to a certain input is the focus of machine learning. If the BCI offers a few choices the subject can pick, the machine learning algorithm has to classify the incoming data. Throughout the years, various classification algorithms have been proposed, each posing their own advantages and disadvantages.

A major problem classifiers are facing is the tendency to over-fit the brain data, because the latter contains thousands of data points. A small amount of trials and large amount of data leads to worse performance on new data, as a result of over-fitting. One way to counter this effect is to use simple linear classifiers which perform surprisingly well, another is to alter or preselect the data to make sure classifiers utilize it in an optimal way.

Another issue concerning classification performance is the varying nature of the human brain. Not only does every person's brain output its own characteristic brain 'signature', even daily changes from one person can be seen in the signals. These inter-subject and inter-session discrepancies pose a challenge for

classification algorithms to cope with the variance in brain features while still maintaining a high classification performance.

2.5 Output

As the system classifies the input and generates a prediction, this information can be used for control. Computer control by a mouse or cursor is a commonly used paradigm for BCI systems. Paralyzed patients can type emails or dial phone numbers with the use of their brain once the system is set up for their needs. Besides computer control, physical devices are a second way to utilize the classified brain signals. Among those are wheelchair navigation or prosthetic arm guidance to extend the restricted abilities of those patients.

3 Steady-State Visually Evoked Potentials

Our experiments involved the use of steady-state visually evoked potentials (SSVEPs), a type of stimulus used in various BCI research. This chapter will explain how SSVEPs are evoked in the human brain and why it is an effective way to control BCIs.

As described by Regan (1977), steady-state evoked potentials belong to the family of evoked potentials. There is a distinction between transient and steady-state evoked potentials. This evoking stimulus can occur through any modularity, i.e. visually, auditory or even somatosensory.

A sudden stimulation, for example a flash, will cause an evoked potential. If the rate of stimulation is low, a transient evoked potential is generated which can be seen as a peak in EEG-recordings within a limited time window after the stimulus presentation. Contrary to event-related potentials, evoked potentials usually show within the first 100ms after stimulation (Effern et al., 2000). A popular and commonly used event-related signal which is elicited when the subject encounters the requested target on a task, is much slower as its positivity peaks around 300ms, hence the name P300.

In situations where the rate of stimulation is high enough, the repetitive peaks being generated will overlap as the one before has not died away. When speaking of steady-state evoked potentials, these overlapping peaks will form a wave-like signal whose frequency is equal to the rate of the stimulation. In short, if you perceive a stimulus which follows a periodic pattern, the same periodic pattern can be seen in brain activity.

In case of steady-state visually evoked potentials (SSVEPs), the stimulus is visually presented. In most common cases this is a LED flickering at a particular frequency, but either CRT or LCD screens have been successfully used as well. A comparison of different visual stimulators has been described by Wu et al. (2008).

As shown by Herrmann (2001), flicker frequencies up to 50 Hz have been proven to evoke a response in the brain. In addition, visual stimulation at a rate of around 10 Hz seems to elicit an enhanced increase in amplitude compared to other frequencies. Besides this phenomenon, stimulation at a particular frequency shows brain activity at the harmonics of the fundamental frequency as well, e.g. 10 Hz will also show a (weaker) peak at 20 Hz and 30 Hz and possibly even a third and fourth harmonic. Additionally, stimulating at higher frequencies shows activity at sub-harmonic rates, e.g. 10 Hz and 15 Hz peaks in response to 30 Hz stimulation.

The main reason for using the naturally occurring steady-state visually evoked potentials is that it is clear where to look for a signal ((Middendorf et al., 2000), which offers better filtering methods. SSVEP BCI systems using up to 48 targets have been used to successfully control an electric apparatus (Gao et al., 2003).

Besides knowing where to find the fundamental frequencies, the harmonics can also be used to improve the 'certainty' of the classifier as detecting an EEG

peak at 20 Hz and a second reduced peak at 40 Hz increases the certainty of actually being stimulated at 20 Hz. Related to the easier and more certain detection of signals is the little training that is required for system operation. For SSVEP BCIs the user only has to visually focus on the desired stimulus to trigger a response.

4 Experiment: robot navigation

4.1 Introduction

This chapter will describe the hardware components, tasks, signal processing pipeline and other details of the first experiment. In conclusion, the results are shared and the main reason for setting up a second experiment will be given in Chapter 4.5.

For this experiment, a driving robot was controlled via a SSVEP brain-computer-interface. The participant that controlled the robot was attached to a brain-reading device. The received brain signals were sent to a computer which in turn pre-processed, extracted features and classified the data.

In this way, it is possible to autonomously control a vehicle by mere brain activation. If the performance of steering a robot with pure mental effort is sufficiently high, this may open doors to new ways for disabled or paralyzed patients to increase their mobility.

Because this brain-computer-interface experiment will follow the SSVEP paradigm, we can use the flicker rate of each LED to determine which direction the participant is looking at. In result, we can use the generated brain signals to navigate the robot in a direction. For a more comprehensive overview of steady state visually evoked potentials, see chapter 3.

For this experiment we wish to find how well a SSVEP-based BCI performs with an interactive, moving robot mounted with stimuli. More specifically:

- How does the distance between the user and the robot affect performance?
- How does the orientation of the stimuli (LEDs) affect performance?

4.2 Requirements & details

Two male participants (average age of 23) took part in this experiment. Both participants had previous experience with BCI controlled systems. The experiment was carried out in a dark room with a small light bulb for convenience (e.g. controlling the system, positioning the robot and being able to record the experiment).

Several devices were needed to run the experiment. First of all, a driving robot was necessary to be controlled by the participant. In this case, a custom-built LEGO robot was used to receives commands via Bluetooth through the computer (Figure 2). This computer is the main link between all connected parts. For reading the brain signals of the participant, we need an EEG reading head-set.

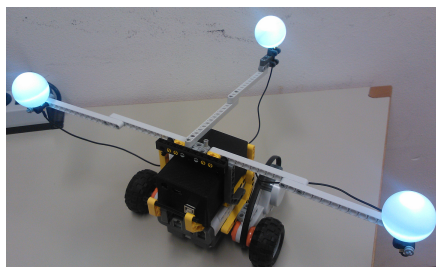


Figure 2: LEGO Mindstorms robot with LEDs for navigation.

When processing signals, the Nyquist rate specifies a theoretically minimum sampling frequency of twice the frequency of the function being sampled without aliasing occurring. Since the participant will only be visually stimulated by three LEDs of various frequencies with a maximum of 23 Hz, the Nyquist rate would be 46 Hz. The Emotiv EEG system, sampling at 128 Hz, is therefore sufficient enough to reliably read the required brain signals. A full-on EEG head cap such as the BioSemi is not needed for this experiment, thus will save us time and effort to set it up. The Emotiv EEG was connected to the computer with the included USB receiver for the necessary data processing.

For the visual stimulation of the participant, a custom-built LED box was used. This box contains an Arduino Uno board which controls three LEDs that each flicker at an individual frequency by using the build-in real time clock. The LEDs were covered by a white diffusor and fitted to a holder.

Because the robot was used for navigating, one important requirement was being wireless. To fulfil this requirement the LED box also contains a lithium-ion battery pack for the necessary power. The complete box is mounted on the driving robot, with each LED at its own position. For an overview of the LED's position, see Figure 3.

The robot used in this experiment is based on the LEGO NXT Mindstorms robotics set. This set contains a NXT brick which holds connections for motor driving plus an additional Bluetooth interface for communicating with the computer. The wireless connection is a major advantage for sending signals when using a mobile device.

For analysis purposes, the complete run for each participant was recorded with a camera that was fixed to the ceiling for a complete view of the track. These video images were streamed to a second computer that saved the complete recording.

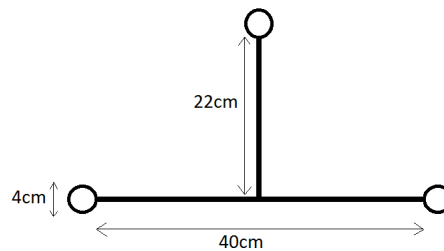


Figure 3: Simplified orientation of the LED holder used in the experiment

4.3 Tasks

At first, the goal and tasks of the experiment were explained to the participant. The first task was needed to calibrate the system. When prompted, the participant has to attend to each LED in turn, to let the system learn which brain signal features belong to which flicker rate.

After calibration, there was a period of 'getting used to' the robot and the BCI system itself. This involved several movement tasks (left, right, forward) as requested by the observer of the experiment. If the participant was confident the final task of driving a complete run to the goal will be smooth, the experiment was continued to the next part.

The next task was a real on-line robot navigation test in which the participant had to follow the predefined path and drive to the goal spot as fast as possible with the least amount of errors. For an overview of the track course that is used for the experiment, see Figure 4. The course is a 2.5m^2 area with a start and goal spot. Between these two, a path with various left and right corners are placed to test the steering ability of the brain-controlled robot. The

will be moving at least several meters in the created area, a wired connection would not be suitable for this purpose.

To connect the LEGO robot to the PC, a connection is made within Windows 7 Professional using the standard Windows Bluetooth drivers.

4.4.2 Initialization

Because Matlab is used for processing the EEG signals, several devices have to be known for Matlab to be able to communicate.

The LEGO robot is both initialized and controlled within Matlab through the RWTH Mindstorms NXT Toolbox¹. This toolbox contains several functions that enables the PC to send commands to the NXT bricks via a USB connection or Bluetooth. Besides controlling the vehicle, the package also includes a Matlab function to open a connection between Matlab and the LEGO NXT brick using a predefined Bluetooth initialization file (.ini) and to return a handle to be used for communication.

For reading brain signals, the Emotiv EEG device is used since it is easy and takes far less time to set up compared to a standard full EEG cap (i.e. Biosemi). To start a connection between the PC and Emotiv, a package called matBCI is used, which can be found here². This bundle of BCI-related framework files contains predefined Emotiv functions to be able to start and initialize the connection between the PC and Emotiv.

4.4.3 Calibration

The majority of BCI systems need a phase to train the parameters of the algorithm. This phase is called calibration and is needed for the correct working of the derived classifications. Calibration is often called the training phase because the brain data gathered during this step is used to alter and improve the algorithm.

For calibration we chose to run an off-screen calibration with the stimuli attached to the robot as this is closer to the real on-line usage scenario as it offers similar lighting conditions, stimulus size and subject's viewing angle to navigation experiment. Off-screen calibration has the most important advantage of not being time-locked to the cues, as stimuli continue to flicker throughout the entire calibration.

To facilitate the user's attention at the right stimulus as requested by the calibration software, auditory cues have been added for each stimulus and another cue to indicate the end of each trial. The on-screen calibration will still run in the background out of sight while the participant attends at the robot's flickering LEDs. Each of the three stimuli/directions will be randomly selected by the system while maintaining an evenly distributed amount of trials among them. The on-screen calibration is done as follows:

¹<http://www.mindstorms.rwth-aachen.de/>

²<http://www.dcc.ru.nl/jdrf/download/>

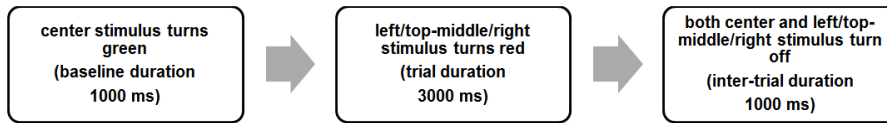


Figure 5: Calibration timeline for each trial. Every direction (left/top-middle/right) will be randomly chosen but ensured the total number of each will be the same. After each trial, the calibration phase will repeat these steps again until all trials are done.

The on-screen calibration is not visible for the participant, only the auditory cues will indicate which stimulus to attend to. In our case the total amount of trials was set at 42, thus 14 trials for each frequency. The trial duration lasted for 3000 milliseconds. Combined with the Emotiv EEG we obtained a total of 384 samples per trial. After every trial an auditory signal informed the participant off-screen about the end of the trial, allowing for a short break between each trial.

4.4.4 Training the classifier

The brain data collected at the calibration phase of the experiment needs to undergo a few changes before it can be used by the classifier. The classifier itself has to derive the correct classification, which in this case is a direction of the robot, from the signal data it is given.

4.4.5 Signal pre-processing

The raw data obtained from the participant by the Emotiv contains a lot of noise, artifacts and possibly electrical grid interference. To counteract these factors and improve the detection of relevant signals, a series of filters and the detection of outliers is applied to the data. This pre-processing step will be done by running the necessary Matlab files from the matBCI package. Accordingly, these functions will modify the EEG data to increase the performance of the classification algorithm. Following is an overview of the pre-processing steps used for this experiment.

4.4.5.1 De-trending

There is a possibility the data picks up (linear) trends that result in the signal comprising a gradual in- or decreasing trend over time. As this trend has no use for our purpose whatsoever, it is highly beneficial for performance aspects to remove the trend from the obtained data. In our case the data is de-trended to achieve an optimal reconstruction of the original signal.

4.4.5.2 Bad channel removal

Electrode channels may receive activity that is not relevant for the system to be processed. These errors can be overcome with the use of a channel filter. Channels with signal values that exceed a predefined value (outliers) will be removed from the data to prevent incorrect training and classification of the algorithm. A value of 3.1 standard deviations above or below average was set as the threshold, every channel exceeding this value was removed from the data.

4.4.5.3 Re-referencing

Electrodes close to one other may have influence on their neighbor's signal characteristics through conduction. To keep this flow of electrical activity between each EEG channel at a minimum, spatial filters are used to maximize the signal strength (each electrode receiving from its own field of interest) and improve signal-to-noise ratio. Another benefit of re-referencing is the decreased correlation between every electrode, so each has its own unique share in signal detection to prevent over-fitting the data.

In our case, two techniques are used to enhance signal quality, namely a Common Average Reference (CAR) filter and a custom SSEP filter. The latter generates a set of spatial filters for a given set of frequencies, such that the power in the chosen frequencies is maximised while the power in all other frequencies is minimised. Thus the spatial filter 'locks-on' to the brain response to the stimulated frequencies. If more than one spatial filter is computed, the best two filters are used for training the classifier.

4.4.5.4 Bad trial removal

Similar to bad channel removal, also trials that contain too many outliers are removed from the training set. The decision whether data is considered an outlier depends on the calculated mean and standard deviation of the data at hand. In our experiment every trial exceeding 3 standard deviations above or below average was rejected from classification.

4.4.5.5 Welch's method

Welch's method Welch (1967) is used to compute the power spectral density (PSD) of the signal. This method is faster and considered better compared to standard FFT-based PSD computation in which the latter may show more noise or spectral leakage in the PSD. The main difference from a standard PSD computation is instead of using a single FFT of all the data, the average absolute FFT for a set of smaller over-lapping time windows is used. In our pre-processing step a Hanning window is used with an overlap of 50%. The width of the window corresponded to 1000 milliseconds of data, rendering a spectrogram resolution of 1 Hz.

4.4.5.6 Frequency range selection

As the stimuli used in our experiments are respectively 13, 17 and 23 Hz, it is known beforehand where a SSVEP response will be. For this reason, only the amplitudes at the fundamental frequencies of 13, 17 and 23 Hz and their the first and second harmonics of each were used for classification.

4.4.6 Classification

To classify the pre-processed data, the Matlab files included in the matBCI package are used. As described above, several filters were used to separate the noise signals from relevant information. In our case, a L2 regularized logistic regression classifier was trained, with parameter tuning using 10-fold cross-validation.

Setting	Driving speed	Steering speed	Turn ratio
Slow	20	25	560
Medium	40	50	560
Fast	60	75	560

Table 1: Robot settings

4.4.7 LEGO NXT robot

Since the classification algorithms will output the detected signal, the result can be used to choose the correct type of movement (left, right, forward). Detecting a signal will result in a command sent to the robot.

The RWTH Mindstorms Toolbox includes motor commands with multiple parameters to be set. An additional file was written to be able to vary the speed of both driving and steering. For an overview of the parameter settings, see Table 1. These parameters can be used when sending the driving command to the NXT robot, with a constant distance travelled per move. For our experiment only the low speed parameters were used, respectively 20 Power synchronized forward drive and 560 degrees Turnratio for both left and right turns. The file also give the option for time-based instead of ratio-based turns, though the latter seemed to provide more accurate results.

4.5 Results

The total experiment consisted of one offline calibration run and an on-line navigation task that was recorded with a camera. The calibration run was carried out with the robot situated in the middle of the track.

Part.	time to finish (min)	# steps	% correct	bits/classf.	bits/sec
1	32:50	142	70.4%	0.41	0.029
2	08:49	83	59.0%	0.20	0.031

Table 2: Results (time to finish, number of steps, percentage correct, bits per classification and bits per second) and information transfer rate for each participant based on Wolpaw’s definition (Wolpaw et al., 1998)

Results of both participants were obtained from the recorded videos (Table 2). Participant one took thirty-two minutes and fifty seconds to finish the track. During the complete run a total of 142 steps were taken from start to finish, of which 100 were correct. This means the correct classification rate was slightly over 70.4%. Results of participant one show a navigation speed of 4.33 steps per minute, with approximately 3.05 steps per minute being as intended by the user to be able to follow the track.

Different results were collected from participant two. Total time from start to finish was recorded as eight minutes and forty-nine seconds. An important remark was that the user took a shortcut, thus bypassed about one-third of the total length of the track. The participant made note of the amount of errors during navigation as being tiresome, as requested a skip of the straight part of the track.

Above explains the shorter time from start to finish. In total a number of eighty-three steps were taken, of which forty-nine were correct. This results in a correct classification rate of 59.0%. As the inter-trial pause was removed during navigation for this participant, the step-speed of the robot was approximately 9.41 steps per minute, with 5.56 correct ones per minute.

$$R = \log^2(N) + P \log^2(P) + (1 - P) \log^2\left(\frac{1 - P}{N - 1}\right) \quad (1)$$

The information transfer rate for both participants was calculated using Wolpaw's definition (Wolpaw et al., 1998) as derived from Pierce (1980), in bits per classification (see Equation 1. Multiplying this value with the classification rate per second (amount of steps) yields an ITR of bits per second. Both participants achieved similar ITR results, despite participant one having a higher information rate per classification. This is caused by the higher classification speed from removal of the inter-trial pause of the experiment with participant two.

Unfortunately, the separate performance rates for every distance (close/medium/far) and direction (left/right/forward) could not be reliably calculated. This was caused by short-cutting of the track by both participants so we lacked measurements for several parts of the experiment. As a result, only the performance rate of the total track was calculated as noted above. Due to this, we were not able to answer our research questions for this experiment.

One final remark for this experiment is the fact that on average it takes two correcting moves for every mistake, as incorrect turns require two corrections and an incorrect forward move requires four corrections. This would essentially mean a performance of less than 50% (a correction on average requiring four moves), makes the robot perform below random chance thus be unusable.

Since this first experiment lacked an answer to our questions, we conducted a second experiment that would answer the question how distance between the user and stimulus relates to the performance of the SSVEP BCI system, see chapter 5.

5 Experiment: stimulation at a distance

5.1 Introduction

As the results of the navigation experiment (chapter 4) did not answer our question, we conducted a second experiment to investigate the relation between the user-stimulus distance and the performance of the SSVEP BCI system.

For this experiment, we focussed on varying the distance while keeping the orientation of the stimuli consistent. The LED holder of the navigation experiment was separated from the robot and used to research the impact of distance between subject and stimulus on the classification performance. EEG-reading was similar to the previous experiment, with a computer processing and classifying the data. The results were analyzed to test whether and from which point the distance affects the performance of a BCI system.

5.2 Requirements & details

Devices used are similar to the navigation experiment, minus the LEGO robot. Again the Emotiv EEG system was used for reading brain signals at 128 Hz sampling rate. The separated LED box of the robot with each light equipped on the holder was used for stimulation. Stimulation was done at respectively 13, 17 and 23 Hz.

The distance of a stimulus in SSVEP BCI experiments is commonly measured in degrees of visual angle, for example see (Resalat & Setarehdan, 2012). This is done to calculate the stimulated retinal area of the subject. For our experiment we followed a similar approach by varying the angle of stimulation between 6 and 16 degrees, in steps of 2 degrees. Because the horizontal distance was varied, we had to convert angle stimulation to distance in centimeters, dependent on the height on the participant. A stimulation angle higher than 16 or lower than 6 degrees could not be achieved as this would result in immeasurable points from the participant. Each participant's individual distance points were calculated and marked on the floor of the room.

5.3 Tasks

Two males and one female (average age was 23.7) participated in the experiment. Both males had experience with BCI systems. The participants were instructed beforehand to follow the auditory cues generated by the calibration phase of the system, see chapter 4.4.3 for more information about the calibration step. Similar to the navigation experiment, the calibration step consisted of 42 trials in total, each stimulus being offered randomly the same amount of times. The inter-trial duration was set at 0 milliseconds as participants indicated no need for an extra break beside the baseline duration of 1000 milliseconds and an additional period of time for the auditory cues lasting between 3 and 4 seconds.

As stated above, the distance was varied from 16 to 6 degrees of visual angle between the two outermost stimuli. This angle relates to a certain horizontal length which was calculated by factoring in the length of the participant. All

six measurements were marked on the floor before the experiment, so it was clear to both participant and observer where to place or look at the LED holder.

5.4 Experiment

For each participant the experiment consisted of six calibrations and their accompanying classifier performances at the varied distances. Because the distance was varied from close to far, fatigue might play a role and influence the EEG readings. To counter this effect, a second run was done (including the six measurements) in a random order. This means a total of twelve measurements were done, of which six of the second run were randomly presented to the participant.

Pre-processing steps were similar to the navigation experiment, i.e. detrending, bad channel removal, re-referencing, bad trial removal, Welch's method and frequency filter at 13, 17 and 23 Hz. The experiment was carried out in a dark room with a small light bulb for convenience (e.g. controlling the system and positioning the LED holder).

5.5 Results

EEG data for each participant was pre-processed and averaged for increasing and random order measurements. With the pre-processing step, spectral power for each frequency was obtained from the power spectral density transform. The data was then divided for each stimulation, i.e. amplitudes of 13 Hz, 17 Hz, and 23 Hz frequencies were selected from each different frequency trial. This way, it is also clear how the amplitudes for the non-target frequencies behave during stimulation of a target frequency.

Every frequency amplitude was then averaged over each frequency trial for each visual angle, yielding the frequency amplitude averages over 42 trials per measurement. Next, the amplitude data of ordered and random measurements was combined and averaged. This means the average frequency power of each frequency stimulation is acquired for every visual angle measured. Finally, for each frequency stimulation the true positive rate is calculated by means of:

$$\text{True positive rate} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \quad (2)$$

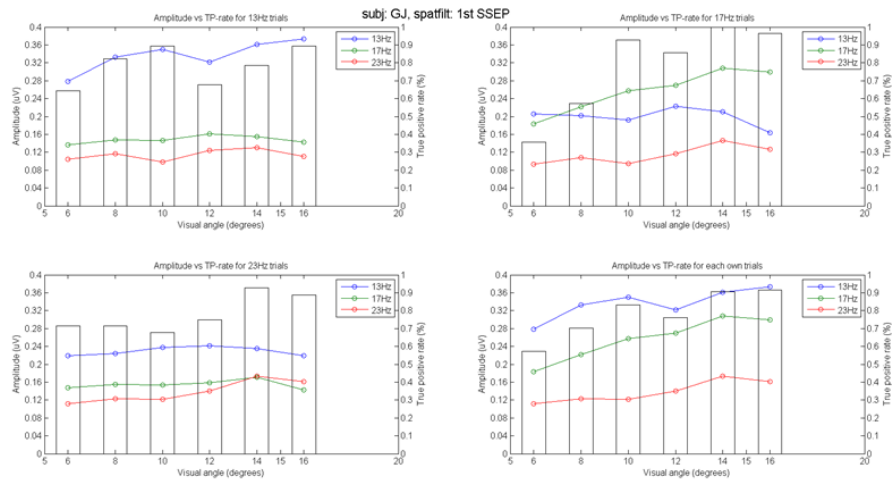


Figure 6: Results for subject G.J. The average power of each amplitude for every frequency is plotted. Each line plot contains the average amplitude measurements for a particular stimulus frequency. The bar plot indicates the true positive rate for every measurement for the frequency stimulated during the particular trials. The bottom-right plot holds the average frequency power of each target's trials. All plots are done with the amplitudes from the first spatial ssep filter, as a result it might be possible plots are different when using the second spatial ssep filter.

For each plot, the target's amplitude correlates significantly ($p < 0.05$) with the visual angle of stimulation, thus distance. Correlations are $R=0.84$; $R=0.80$; $R=0.92$ respectively for each target's frequency. An overall view of all plots shows that a 13 Hz stimulation yields the highest amplitudes for this subject.

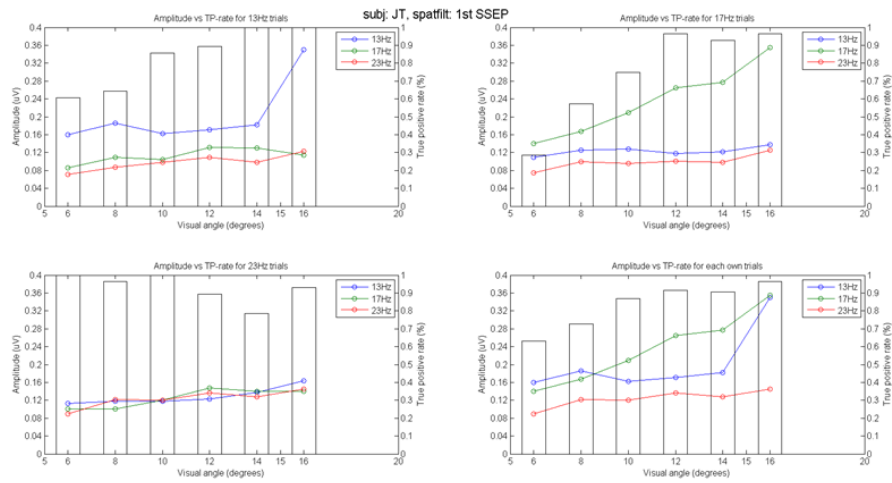


Figure 7: Results for subject JT. The average power of each amplitude for every frequency is plotted. Each line plot contains the average amplitude measurements for a particular stimulus frequency. The bar plot indicates the true positive rate for every measurement for the frequency stimulated during the particular trials. The bottom-right plot holds the average frequency power of each target's trials. All plots are done with the amplitudes from the first spatial ssep filter, as a result it might be possible plots are different when using the second spatial ssep filter.

Surprisingly, for 13 Hz stimulation only the 23 Hz amplitudes correlate for this subject ($R=0.91$, $p<0.05$). For 17 Hz stimulation, both 17 Hz and 23 Hz amplitudes correlate with the visual angle ($R=0.99$, $p<0.01$; $R=0.85$, $p<0.05$). When stimulating at 23 Hz, all three frequency amplitudes correlate significantly, albeit very similar ($R=0.89$, $R=0.88$, $R=0.87$). For this subject it seems that a 17 Hz gives the best results in terms of frequency power.

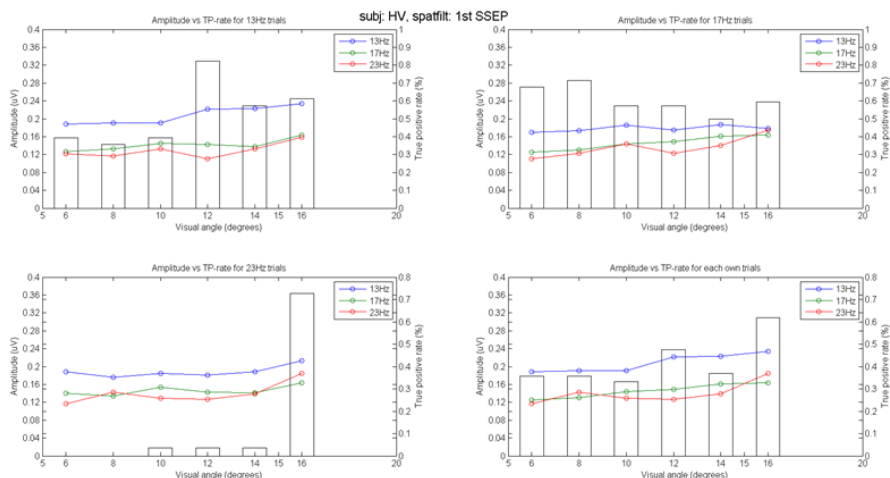


Figure 8: Results for subject HV. The average power of each amplitude for every frequency is plotted. Each line plot contains the average amplitude measurements for a particular stimulus frequency. The bar plot indicates the true positive rate for every measurement for the frequency stimulated during the particular trials. The bottom-right plot holds the average frequency power of each target’s trials. All plots are done with the amplitudes from the first spatial ssep filter, as a result it might be possible plots are different when using the second spatial ssep filter.

For 13 Hz, a correlation between amplitude and visual angle is found for 13 Hz and 17 Hz power ($R=0.93$, $p<0.01$; $R=0.83$, $p<0.05$ respectively). For 17 Hz, correlations are found for 17 Hz and 23 Hz amplitudes ($R=0.99$, $p<0.01$; $R=0.83$, $p<0.05$). Stimulating at 23 Hz no significant relationship is found between the visual angle of stimulation and frequency power.

Figures of the plots show the spectral power of the target’s frequency decreases over distance, as a smaller visual angle means a farther distance. However, the remaining two non-target frequencies remain reliably constant over distance.

The true positive rate indicated in the plots seems to be reliant on the relative difference in power between the target and non-target frequencies. A correlation analysis showed this is not uniformly the case for all participants. See Table tables 3 to 5 for an overview of correlations between the positive hitrate and frequency amplitude and residuals. Residuals were calculated by subtracting the maximum of either non-target frequency amplitude from the target’s frequency amplitude.

Subject GJ	Hitrate 13 Hz trials	Hitrate 17 Hz trials	Hitrate 23 Hz trials
Amplitude	$R=0.89$, $p<0.05$	$R=0.96$, $p<0.01$	$R=0.86$, $p<0.05$
Residuals	$R=0.91$, $p<0.05$	$R=0.90$, $p<0.05$	$R=0.92$, $p<0.01$

Table 3: Correlations for subject GJ. Correlation tests were done between positive hitrate and the frequency amplitudes of targets and the difference between the target’s amplitude and other’s. Note: n.s. means no significant correlation was found.

Subject JT	Hitrates 13 Hz trials	Hitrates 17 Hz trials	Hitrates 23 Hz trials
Amplitude	R=0.49, n.s.	R=0.89, p<0.05	R=-0.47, n.s.
Residuals	R=0.32, n.s.	R=0.89, p<0.05	R=0.15, n.s.

Table 4: Correlations for subject JT. Correlation tests were done between positive hitrate and the frequency amplitudes of targets and the difference between the target's amplitude and other's. Note: n.s. means no significant correlation was found.

Subject HV	Hitrates 13 Hz trials	Hitrates 17 Hz trials	Hitrates 23 Hz trials
Amplitude	R=0.82, p<0.05	R=-0.82, p<0.05	R=0.92, p<0.01
Residuals	R=0.75, n.s.	R=-0.61, n.s.	R=0.64, n.s.

Table 5: Correlations for subject HV. Correlation tests were done between positive hitrate and the frequency amplitudes of targets and the difference between the target's amplitude and other's. Note: n.s. means no significant correlation was found.

6 Conclusion - Discussion

This work involved two separate experiments, each testing its own field of interest. The navigation experiment showed how a distant SSVEP-controlled robot could achieve classification rates between 59.0%-70.4%, accompanying an ITR of approximately 0.03 bits per second. This classification rate is very reasonable compared to other BCI systems. Questionnaires have demonstrated that motion-disabled BCI users, such as those suffering from ALS or tetraplegia, do not mind a slow or non-perfect system. Time not being an issue is one of their arguments, as they are pleased to be able to interact with their friends, family and environment.

However, given it takes a minimum of four corrections for an incorrect forward move and a minimum of one for an incorrect turn, a mistake takes on average two correction moves. This would mean a performance of less than 50% effectively results in performance below random chance.

The second experiment attained results of how SSVEP frequency power relates to the distance, thus visual angle, of stimulation. True positive rates of the trials fluctuate between different subjects, indicating the generally influencing inter-subject difference of a BCI system. As the results between participants are quite variable, it is hard to draw a solid conclusion. The majority of the found correlations do indicate a relation between the spectral power of the frequency and the derived positive hit rate.

One final surprising result from the second experiment was the relatively constant amplitude of both non-target frequency during target stimulation at all visual angles. A suggestion for the cause of this effect could be due to the fact the decreased size over distance is offset by the increased sensor density (more foveal area stimulated) as the target moves towards the subject. This indicates non-target stimuli have only a small role in signal detection, which means it is easier to detect the attended stimuli and have a better operating BCI system.

6.1 Improvements and future work

Taking our current experiment components into account, several improvements can be made for future research. One obvious change would be to use an improved EEG-cap for signal detection. A larger amount of electrodes will have a better spatial resolution. This will improve signal-to-noise ratio and therefore increase the classifier performance of the BCI system.

Second, the experiments show a lot of difference between the participants in terms of classification rate and performance. Employing more participants for next experiments will evenly distribute and average results to be able to draw solid conclusions.

Third, as the second experiment only took the distance of stimuli into account without varying the orientation, we ought to conduct a new experiment that investigates the influence of orientation on SSVEP BCI performance. Most probably, when orienting the LED holder either left or right the stimuli are aligned which might decrease the classifier's accuracy.

Next, results from the second experiment suggest participants are suspect to predominantly lower stimulation frequencies that render higher frequency

amplitudes. Several plots show a high frequency amplitude of 13 Hz despite it not being stimulated at those particular trials. As shown by Regan (1989) a low frequency of 10 Hz excites the largest SSVEP amplitude followed by 16Hz-18Hz. This would mean a change to stimulation frequencies could improve signal detection.

Finally, future work may research the relationship between the amount of stimulated retinal cones and frequency amplitude. Figure 9 depicts the relative visual sharpness for a left human eye measured from the fovea and shows a steep decline in acuity outwards.

The stimulated foveal surface gives rise to cortical magnification of sensory processing, which leads to increased signal power. If the frequency power of stimulation is indeed dependent on the amount of stimulated retinal cones in the fovea, this may offer possibilities for new BCI techniques.

This means if you assure two separate stimuli do not converge in the fovea, signal detection might improve.

Hopefully our research is a stepping stone for others to continue the exploration of SSVEP use for robot navigation.

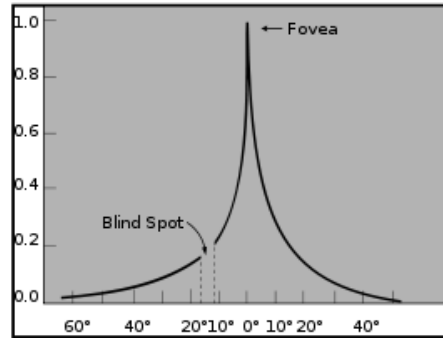


Figure 9: Relative acuity of human left eye in degrees from the fovea (Hunziker, 2006).

7 Acknowledgements

As a lot of scientific research includes the collaboration of people, this thesis would not be possible without the help of others.

First of all I would like to thank my supervisors Louis Vuurpijl and Jason Farquhar for the necessary guidance throughout this past period of time. Next, the Technical Support Group deserves appreciation for building the LED box which was used many times during the experiments, but also for helping out with the several hardware requests from my side.

Also, I am grateful for Loukianos Spyrou and especially Jason Farquhar for helping me out numerous times with BCI/Matlab-related issues when processing brain data. Besides, I would like to thank everyone else for the fruitful discussions we had and suggestions given, in particular Jordy Thielen for sharing his knowledge of subjects ranging from EEG peculiarities to LaTeX tips.

Finally, many thanks to my family and friends for showing an interest in my project throughout this period, what proved to be a huge motivational help for finishing my thesis.

8 References

- Blankertz, B., Sannelli, C., Halder, S., Hammer, E. M., Kübler, A., Müller, K.-R., et al. (2010). Neurophysiological predictor of smr-based bci performance. *NeuroImage*, 51(4), 1303–1309.
- Conradi, J., Blankertz, B., Tangermann, M., Kunzmann, V., & Curio, G. (2009). Brain-computer interfacing in tetraplegic patients with high spinal cord injury. *Int J Bioelectromagnetism Volume*, 11(2), 65–68.
- Effern, A., Lehnertz, K., Schreiber, T., Grunwald, T., David, P., & Elger, C. (2000). Nonlinear denoising of transient signals with application to event-related potentials. *Physica D: Nonlinear Phenomena*, 140(3), 257–266.
- Farquhar, J., Blankespoor, J., Vlek, R., & Desain, P. (2008). Towards a noise-tagging auditory bci-paradigm. In *Proceedings of the 4th international brain-computer interface workshop and training course* (pp. 50–55).
- Gao, X., Xu, D., Cheng, M., & Gao, S. (2003). A bci-based environmental controller for the motion-disabled. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 11(2), 137–140.
- Herrmann, C. S. (2001). Human eeg responses to 1–100 hz flicker: resonance phenomena in visual cortex and their potential correlation to cognitive phenomena. *Experimental brain research*, 137(3-4), 346–353.
- Hunziker, H. W. (2006). *Im auge des lesers: vom buchstabieren zur lesefreude: foveale und periphere wahrnehmung*. transmedia verlag.
- Kelly, S. P., Lalor, E. C., Reilly, R. B., & Foxe, J. J. (2005). Visual spatial attention tracking using high-density ssvep data for independent brain-computer communication. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 13(2), 172–178.
- Lalor, E. C., Kelly, S. P., Finucane, C., Burke, R., Smith, R., Reilly, R. B., et al. (2005). Steady-state vep-based brain-computer interface control in an immersive 3d gaming environment. *EURASIP journal on applied signal processing*, 2005, 3156–3164.
- Leeb, R., Friedman, D., Slater, M., & Pfurtscheller, G. (2007). A tetraplegic patient controls a wheelchair in virtual reality. In *Brainplay 07 brain-computer interfaces and games workshop at ace (advances in computer entertainment) 2007* (p. 37).
- Lins, O. G., & Picton, T. W. (1995). Auditory steady-state responses to multiple simultaneous stimuli. *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section*, 96(5), 420–432.
- Middendorf, M., McMillan, G., Calhoun, G., & Jones, K. S. (2000). Brain-computer interfaces based on the steady-state visual-evoked response. *Rehabilitation Engineering, IEEE Transactions on*, 8(2), 211–214.

- Müller, M. M., Picton, T. W., Valdes-Sosa, P., Riera, J., Teder-Sälejärvi, W. A., & Hillyard, S. A. (1998). Effects of spatial selective attention on the steady-state visual evoked potential in the 20–28 hz range. *Cognitive Brain Research*, *6*(4), 249–261.
- Muller-Putz, G., Scherer, R., Neuper, C., & Pfurtscheller, G. (2006). Steady-state somatosensory evoked potentials: suitable brain signals for brain-computer interfaces? *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *14*(1), 30–37.
- Muller-Putz, G. R., & Pfurtscheller, G. (2008). Control of an electrical prosthesis with an ssvep-based bci. *Biomedical Engineering, IEEE Transactions on*, *55*(1), 361–364.
- Ortner, R., Allison, B. Z., Korisek, G., Gaggl, H., & Pfurtscheller, G. (2011). An ssvep bci to control a hand orthosis for persons with tetraplegia. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *19*(1), 1–5.
- Patel, A. D., & Balaban, E. (2004). Human auditory cortical dynamics during perception of long acoustic sequences: phase tracking of carrier frequency by the auditory steady-state response. *Cerebral Cortex*, *14*(1), 35–46.
- Pfurtscheller, G., Solis-Escalante, T., Ortner, R., Linortner, P., & Muller-Putz, G. R. (2010). Self-paced operation of an ssvep-based orthosis with and without an imagery-based "brain switch:" a feasibility study towards a hybrid bci. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *18*(4), 409–414.
- Pierce, J. R. (1980). An introduction to information theory: Symbols, signals and noise author: John r. pierce, publisher: Dover publications p.
- Regan, D. (1977). Steady-state evoked potentials. *JOSA*, *67*(11), 1475–1489.
- Regan, D. (1989). Human brain electrophysiology.
- Resalat, S. N., & Setarehdan, S. K. (2012). A study on the effect of the inter-sources distance on the performance of the ssvep-based bci systems. *American Journal of Biomedical Engineering*, *2*(1), 24–31.
- Silvoni, S., Cavinato, M., Volpato, C., Ruf, C. A., Birbaumer, N., & Piccione, F. (2013). Amyotrophic lateral sclerosis progression and stability of brain-computer interface communication. *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*(0), 1–7.
- Silvoni, S., Volpato, C., Cavinato, M., Marchetti, M., Priftis, K., Merico, A., et al. (2009). P300-based brain-computer interface communication: evaluation and follow-up in amyotrophic lateral sclerosis. *Frontiers in neuroscience*, *3*.
- van Gerven, M., Farquhar, J., Schaefer, R., Vlek, R., Geuze, J., Nijholt, A., et al. (2009). The brain-computer interface cycle. *Journal of Neural Engineering*, *6*(4), 041001.
- Welch, P. (1967). The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *Audio and Electroacoustics, IEEE Transactions on*, *15*(2), 70–73.

- Wolpaw, J. R., Ramoser, H., McFarland, D. J., & Pfurtscheller, G. (1998). Eeg-based communication: improved accuracy by response verification. *Rehabilitation Engineering, IEEE Transactions on*, *6*(3), 326–333.
- Wu, Z., Lai, Y., Xia, Y., Wu, D., & Yao, D. (2008). Stimulator selection in ssvep-based bci. *Medical engineering & physics*, *30*(8), 1079–1088.

9 Appendix

9.1 Robot files

```
1 [Bluetooth]
2
3 SerialPort=COM40
4 BaudRate=9600
5 DataBits=8
6
7 SendSendPause=5
8 SendReceivePause=25
9
10 Timeout=2
```

```
1 %% closeHandle.m
2 %% closing all nxt handles
3
4 COM_CloseNXT all
5 disp('Closed handle')
```

```
1 %% openBT.m
2 %% Open new BT connection
3
4 % look for USB devices, THEN for Bluetooth
5 global handleNXT;
6 handleNXT = COM_OpenNXT('bluetooth_.ini');
7
8 % set global default handle
9 COM_SetDefaultNXT(handleNXT);
10
11 disp('====Bluetooth opened====');
```

```
1 function [drvSpd trnSpd] = getParam( speed )
2 %%function [drvSpd trnSpd] = getParam( speed )
3 %
4 % returns parameters for driving speed and steering by giving a ...
5 % speed ('low', 'med', 'high')
6
7 switch speed
8     case 'low'
9         drvSpd = 20;
10        trnSpd = 25;
11
12     case 'med'
13         drvSpd = 40;
14         trnSpd = 50;
15
16     case 'high'
17         drvSpd = 60;
18         trnSpd = 75;
19     otherwise
20         return;
```

```
21 end
```

```
1 function [] = sendDriveCommand( direction, speed )
2 %%function [] = sendDriveCommand( direction, speed )
3 %
4 % direction = 'f' for forward, 'l' for left, 'r' for right, 'u' ...
   for u-turn
5 % speed ('low', 'med', 'high') determines the speed at which the ...
   robot drives/turns
6
7 [drvSpd trnSpd] = getParam(speed); % get the drive/turn parameters
8 avgDrvSpd = 40; % average drive speed (20, 40, 60)
9 avgTrnSpd = 50; % average turn speed (25, 50, 75)
10
11 % set the drive and steer duration to compensate for speed ...
   (higher speed = less
12 % duration for equal distance per command
13   %drvDur = avgDrvSpd / drvSpd;
14   %trnDur = avgTrnSpd / trnSpd;
15   %trnDur = avgTrnSpd / (trnSpd * 1.3); % 45 degrees turn
16   %trnDur = (2*avgTrnSpd) / (trnSpd * 1.3); % 90 degrees turn
17   drvDur = avgDrvSpd / (drvSpd * 2);
18   turnRatio = 560;
19
20 %% Make a move dependent on direction input
21 switch direction
22     case 'f' % driving forward
23         disp('====Forward====')
24         % reset sync counters
25         NXT_ResetMotorPosition(MOTOR_A, true);
26         NXT_ResetMotorPosition(MOTOR_B, true);
27
28         % drive forward in sync
29         DirectMotorCommand(MOTOR_A, drvSpd, 0, 'off', ...
30             MOTOR_B, 0, 'off');
31         pause(drvDur*1.75); % custom parameter to fit the track
32         StopMotor('all', 'on');
33
34         % reset sync counters
35         NXT_ResetMotorPosition(MOTOR_A, true);
36         NXT_ResetMotorPosition(MOTOR_B, true);
37     case 'l' % go left approx 45 degrees (or 90, see above ...
38         for param)
39         disp('====Left====')
40
41         %—Code used for degree base turning (more precise)
42         DirectMotorCommand(MOTOR_A, drvSpd, turnRatio, 'on', ...
43             'off', 0, 'off'); % turn for 560 deg, approx 90 ...
44             degrees
45
46         %—Code used for time based turning—
47         %StopMotor(MOTOR_B, 'on'); % active break on wheel
48         %DirectMotorCommand(MOTOR_A, trnSpd, 0, 'off', ...
49             'off', 0, 'off');
50         %pause(1);
51         %pause(trnDur);
52         %StopMotor(MOTOR_A, 'on');
53         % reset sync counters
54         %NXT_ResetMotorPosition(MOTOR_B, true);
55         %NXT_ResetMotorPosition(MOTOR_A, true);
```

```

52
53     case 'r' % go right approx 45 degrees (or 90, see above ...
          for param)
54         disp('====Right====')
55
56         %—Code used for degree base turning (more precise)—
57         DirectMotorCommand(MOTOR_B, drvSpd, turnRatio, 'on', ...
          'off', 0, 'off'); % turn for 560 deg
58
59         %—Code used for time based turning—————
60         %StopMotor(MOTOR_A, 'on'); % active break on wheel
61         %DirectMotorCommand(MOTOR_B, trnSpd, 0, 'off', ...
          'off', 0, 'off');
62         %pause(1);
63         %pause(trnDur);
64         %StopMotor(MOTOR_B, 'on');
65         % reset sync counters
66         %NXT_ResetMotorPosition(MOTOR_A, true);
67         %NXT_ResetMotorPosition(MOTOR_B, true);
68
69     case 'u' % make a u-turn full speed (approx 180 degrees)
70         StopMotor(MOTOR_B, 'on'); % active break on wheel
71         DirectMotorCommand(MOTOR_A, 100, 0, 'off', 'off', 0, ...
          'off');
72         pause(1.5);
73         disp('====U-turn====')
74
75         StopMotor(MOTOR_A, 'off');
76         StopMotor(MOTOR_B, 'off');
77
78         % reset sync counters
79         NXT_ResetMotorPosition(MOTOR_A, true);
80         NXT_ResetMotorPosition(MOTOR_B, true);
81
82     otherwise % wrong input, stop.
83         StopMotor(MOTOR_A, 'off');
84         StopMotor(MOTOR_B, 'off');
85         % reset sync counters
86         NXT_ResetMotorPosition(MOTOR_A, true);
87         NXT_ResetMotorPosition(MOTOR_B, true);
88         disp('====Wrong input, program quit====')
89     return
90 end
91
92 end

```

```

1 %% resetAll.m
2 %% close all nxt handles and clear variables
3
4 COM_CloseNXT('all')
5 close all
6 clear all
7 disp('Reset finished')

```

9.2 Buffer_BCI adaptations

```
1 %% configureIM.m
2 % guard to prevent running multiple times
3 if ( exist('imConfig','var') && ~isempty(imConfig) ) return; end;
4 imConfig=true;
5
6 run ../utilities/initPaths;
7
8 buffhost='localhost';buffport=1972;
9 global ft_buff; ft_buff=struct('host',buffhost,'port',buffport);
10 % wait for the buffer to return valid header information
11 hdr=[];
12 while ( isempty(hdr) || ~isstruct(hdr) || (hdr.nchans==0) ) % ...
13     wait for the buffer to contain valid data
14     try
15         hdr=buffer('get_hdr',[],buffhost,buffport);
16     catch
17         hdr=[];
18         fprintf('Invalid header info... waiting.\n');
19     end;
20     pause(1);
21 end;
22
23 % set the real-time-clock to use
24 initgetwTime();
25 initsleepSec();
26
27 %capFile='cap_tmsi_mobita_im';
28 capFile='1010';
29 %capFile='cap_tmsi_mobita_black';
30 %capFile='cap_tmsi_mobita_num';
31 %capFile='cap_tmsi_mobita_p300';
32
33 verb=0;
34 buffhost='localhost';
35 buffport=1972;
36 nSyms=3;
37 nSeq=42; % minimum 12 nSeq
38 nBlock=2;%10; % number of stim blocks to use
39 trialDuration=3;
40 baselineDuration=1;
41 intertrialDuration=0;
42 feedbackDuration=1; % used for epoch-feedback
43 moveScale = .1;
44 bgColor=[.5 .5 .5];
45 fixColor=[1 0 0];
46 tgtColor=[0 1 0];
47
48 trlen_ms=trialDuration*1000;
49 trlen_ms_ol=1000; % trial length for classifier predictions in ...
50 on-line (testing)
51
52 %% sound files for calibration
53 [leftsound,audioSampleRate,~] = ...
54     wavread('../imaginedMovement/media/leftsound.wav');
55 [rightsound,~,~] = ...
56     wavread('../imaginedMovement/media/rightsound.wav');
57 [upsound,~,~] = wavread('../imaginedMovement/media/upsound.wav');
58 [endsound,~,~] = wavread('../imaginedMovement/media/endsound.wav');
```

```

55 calib_sounds = [rightsound(:,1), upsound(:,1), leftsound(:,1)]; ...
    % single channel

```

```

1 %% imCalibrateStimulus.m
2 if ( ~exist('imConfig','var') || ~imConfig ) configureIM(); end;
3
4 % wait for the buffer to return valid header information
5 hdr=[];
6 while ( isempty(hdr) || ~isstruct(hdr) || (hdr.nchans==0) ) % ...
    wait for the buffer to contain valid data
7     try
8         hdr=buffer('get_hdr',[],buffhost,buffport);
9     catch
10        hdr=[];
11        fprintf('Invalid header info... waiting.\n');
12    end;
13    pause(1);
14 end;
15
16 % make the target sequence
17 tgtSeq=mkStimSeqRand(nSyms,nSeq);
18
19 % make the stimulus
20 %figure;
21 fig=gcf;
22 set(fig,'Name','Imagined Movement','color',[0 0 ...
23        0],'menubar','none','toolbar','none','doublebuffer','on');
24 clf;
25 ax=axes('position',[0.025 0.025 .95 .95], 'units','normalized', ...
26        'visible','off', 'box','off', ...
27        'xtick',[], 'xticklabelmode','manual', 'ytick',[], ...
28        'yticklabelmode','manual',...
29        'color',[0 0 0], 'DrawMode','fast', ...
30        'nextplot','replacechildren',...
31        'xlim',[-1.5 1.5], 'ylim',[-1.5 1.5], 'Ydir','normal');
32
33 stimPos=[]; h=[];
34 stimRadius=.5;
35 theta=linspace(0,pi,nSyms); stimPos=[cos(theta);sin(theta)];
36 for hi=1:nSyms;
37     h(hi)=rectangle('curvature',[1 1], ...
38        'position',[stimPos(:,hi)-stimRadius/2;stimRadius*[1;1]], ...
39        'facecolor',bgColor);
40 end;
41
42 % add symbol for the center of the screen
43 stimPos(:,nSyms+1)=[0 0];
44 h(nSyms+1)=rectangle('curvature',[1 1], ...
45        'position',[stimPos(:,nSyms+1)-stimRadius/4;stimRadius/2 ...
46        *[1;1]], ...
47        'facecolor',bgColor);
48 set(gca,'visible','off');
49
50 % play the stimulus
51 % reset the cue and fixation point to indicate trial has finished
52 set(h(:),'facecolor',bgColor);
53 sendEvent('stimulus.training','start');
54 drawnow; pause(5); % N.B. pause so fig redraws
55
56 for si=1:nSeq;
57     if ( ~ishandle(fig) ) break; end;

```

```

51     sleepSec(intertrialDuration);
52     % show the screen to alert the subject to trial start
53     set(h(end),'facecolor',fixColor); % red fixation indicates ...
54         trial about to start/baseline
55     drawnow;% expose; % N.B. needs a full drawnow for some reason
56     sendEvent('stimulus.baseline','start');
57     sleepSec(baselineDuration);
58     sendEvent('stimulus.baseline','end');
59
60     % show the target
61     fprintf('%d\tgt=%d : ',si,find(tgtSeq(:,si)>0));
62
63     %%%% this play the sound cue at the start of a trial
64     sound(calib_sounds(:, find(tgtSeq(:,si)>0)),audioSampleRate); ...
65         %give auditory indication of fixation
66
67     set(h(tgtSeq(:,si)>0),'facecolor',tgtColor);
68     set(h(tgtSeq(:,si)<=0),'facecolor',bgColor);
69     set(h(end),'facecolor',[0 1 0]); % green fixation indicates ...
70         trial running
71     sendEvent('stimulus.target',find(tgtSeq(:,si)>0));
72     drawnow;% expose; % N.B. needs a full drawnow for some reason
73     sendEvent('stimulus.trial','start');
74     % wait for trial end
75     sleepSec(trialDuration);
76
77     % reset the cue and fixation point to indicate trial has finished
78     set(h(:),'facecolor',bgColor);
79     drawnow;
80     sendEvent('stimulus.trial','end');
81
82     %%%% this plays the sound cue at the end of a trial
83     sound(endsound, audioSampleRate); % give auditory indication of ...
84         end of trial
85
86     ftime=getwTime();
87     fprintf('\n');
88 end % sequences
89 % end training marker
90 sendEvent('stimulus.training','end');
91
92 % thanks message
93 text(mean(get(ax,'xlim')),mean(get(ax,'ylim')),{'That ends the ...
94     training phase.','Thank you for your ...
95     patience'},'HorizontalAlignment','center','color',[0 1 ...
96     0],'fontunits','normalized','FontSize',.1);
97 pause(3);

```

```

1 % imEpochFeedbackStimulus.m
2 configureIM();
3
4 % make the target sequence
5 tgtSeq=mkStimSeqRand(nSyms,nSeq);
6
7 % make the stimulus display
8 fig=gcf;
9 clf;
10 set(fig,'Name','Imagined Movement','color',[0 0 0], ...
11     'menubar','none','toolbar','none','doublebuffer','on');
12 ax=axes('position',[0.025 0.025 .95 .95],'units','normalized', ...

```

```

        'visible','off', 'box','off',...
12         'xtick',[], 'xticklabelmode','manual', 'ytick',[], ...
            'yticklabelmode','manual',...
13         'color',[0 0 0], 'DrawMode','fast','nextplot', ...
            'replacechildren',...
14         'xlim',[-1.5 1.5], 'ylim',[-1.5 1.5],'Ydir','normal');
15
16 stimPos=[]; h=[];
17 stimRadius=.5;
18 theta=linspace(0,pi,nSyms); stimPos=[cos(theta);sin(theta)];
19 for hi=1:nSyms;
20     h(hi)=rectangle('curvature',[1 ...
        1], 'position',[stimPos(:,hi)-stimRadius/2;stimRadius*[1;1]],...
21         'facecolor',bgColor);
22 end;
23 % add symbol for the center of the screen
24 stimPos(:,nSyms+1)=[0 0];
25 h(nSyms+1)=rectangle('curvature',[1 ...
        1], 'position',[stimPos(:,end)-stimRadius/4;stimRadius/2*[1;1]],...
26         'facecolor',bgColor);
27 set(gca, 'visible','off');
28
29
30 % play the stimulus
31 % reset the cue and fixation point to indicate trial has finished
32 set(h(:), 'facecolor',bgColor);
33 sendEvent('stimulus.testing','start');
34 drawnow; pause(5); % N.B. pause so fig redraws
35 endTesting=false; dvs=[];
36 for si=1:nSeq;
37
38     if ( ~ishandle(fig) || endTesting ) break; end;
39
40     sleepSec(intertrialDuration);
41     % show the screen to alert the subject to trial start
42     set(h(:), 'facecolor',bgColor);
43     set(h(end), 'facecolor',fixColor); % red fixation indicates ...
        trial about to start/baseline
44     drawnow;% expose; % N.B. needs a full drawnow for some reason
45     sendEvent('stimulus.baseline','start');
46     sleepSec(baselineDuration);
47     sendEvent('stimulus.baseline','end');
48
49     % show the target
50     fprintf('%d) tgt=%d : ',si,find(tgtSeq(:,si)>0));
51
52     set(h(tgtSeq(:,si)>0), 'facecolor',tgtColor);
53     set(h(tgtSeq(:,si)<=0), 'facecolor',bgColor);
54     set(h(end), 'facecolor',tgtColor); % green fixation indicates ...
        trial running
55     drawnow;% expose; % N.B. needs a full drawnow for some reason
56     sendEvent('stimulus.target',find(tgtSeq(:,si)>0));
57     sendEvent('stimulus.trial','start');
58
59     % initial fixation point position
60     dvs(:)=0; nPred=0; state=[];
61     trlStartTime=getwTime();
62     timetogo = trialDuration;
63     while (timetogo>0)
64         timetogo = trialDuration - (getwTime()-trlStartTime); % time ...
            left to run in this trial
65         % wait for events to process *or* end of trial *or* out of time

```

```

66     [dat,events,state]=buffer_waitData(buffhost, buffport, ...
        state, 'exitSet',{timetogo*1000 {'stimulus.prediction' ...
        'stimulus.testing'}}, 'verb',verb);
67     for ei=1:numel(events);
68         ev=events(ei);
69         if ( strcmp(ev.type,'stimulus.prediction') )
70             pred=ev.value;
71             % now do something with the prediction....
72             if ( numel(pred)==1 )
73                 if ( pred>0 && pred<=nSyms && isinteger(pred) ) % ...
                    predicted symbol, convert to dv equivalent
74                     tmp=pred; pred=zeros(nSyms,1); pred(tmp)=1;
75                 else % binary problem, convert to per-class
76                     pred=[pred -pred];
77                 end
78             end
79             nPred=nPred+1;
80             dvs(:,nPred)=pred;
81             if ( verb>=0 )
82                 fprintf('dv:');fprintf('%5.4f ',pred);fprintf('\n');
83             end;
84             elseif ( strcmp(ev.type,'stimulus.testing') )
85                 endTesting=true; break;
86             end % prediction events to processa
87         end % if feedback events to process
88         if ( endTesting ) break; end;
89     end % loop accumulating prediction events
90
91     % give the feedback on the predicted class
92     dv = sum(dvs,2); prob=1./(1+exp(-dv)); prob=prob./sum(prob);
93     if ( verb>=0 )
94         fprintf('dv:');fprintf('%5.4f ...
            ',pred);fprintf('\t\tProb:');fprintf('%5.4f ...
            ',prob);fprintf('\n');
95     end;
96
97     [maxVal,predTgt]=max(dv); % prediction is max classifier output
98     acc = prob(predTgt);
99     fprintf('CLASSIFIED AS: %f , %f \n', predTgt, acc);
100    %%%% this plays the sound cue at the start of a trial
101    %sound(calib_sounds(:, ...
        find(tgtSeq(:,si)>0)),audioSampleRate); %give auditory ...
        indication of fixation
102
103    %==ROBOT DRIVING PART=====%%
104    %if acc > 0.8 % sufficiently accurate
105    fprintf('Movement ');
106    switch predTgt % decide which movement to take
107    case 1 % right
108        sendEvent('robot.cmd','right');
109        %sendDriveCommand('r', 'low');
110        fprintf('TURN RIGHT\n');
111    case 2 % forward
112        sendEvent('robot.cmd','forward');
113        %sendDriveCommand('f', 'low');
114        fprintf('DRIVE FORWARD\n ');
115    case 3 % left
116        sendEvent('robot.cmd','left');
117        %sendDriveCommand('l', 'low');
118        fprintf('TURN LEFT\n');
119    end
120    %end

```

```

121     %=====%%
122
123     set(h(:), 'facecolor',bgColor);
124     set(h(predTgt), 'facecolor',tgtColor);
125     drawnow;
126     sendEvent('stimulus.predTgt',predTgt);
127     sleepSec(feedbackDuration);
128
129     % reset the cue and fixation point to indicate trial has finished
130     set(h(:), 'facecolor',bgColor);
131     % also reset the position of the fixation point
132     drawnow;
133     sendEvent('stimulus.trial', 'end');
134
135     %%%% this plays the sound cue at the end of a trial
136     %sound(endsound, audioSampleRate); % give auditory ...
        indication of end of trial
137
138     ftime=getwTime();
139     fprintf('\n');
140 end % loop over sequences in the experiment
141 % end training marker
142 sendEvent('stimulus.testing', 'end');
143 text(mean(get(ax, 'xlim')),mean(get(ax, 'ylim')),{'That ends the ...
        testing phase.', 'Thanks for your ...
        patience'}, 'HorizontalAlignment', 'center', 'color', [0 1 ...
        0], 'fontunits', 'normalized', 'FontSize', .1);
144 pause(3);

```

```

1 %% startRobotBuffer.m
2 %% Separate buffer listening for Robot command events
3
4 openBT;
5
6 % adapted from buffer_waitData help
7 state=[];
8 while ( true )
9     % block until we've got new events
10    [data, devents, state]=buffer_waitData([], [], state, ...
        'exitSet', {'robot.cmd', 'cmd'});
11    if ( any(matchEvents(devents, 'cmd', 'exit')) ) % check for ...
        exit events, stop if found
12    break;
13    end
14    % process the new (non-exit) events we've got
15    if ( ~isempty(devents) )
16        fprintf('not empty\n');
17        if (strcmp(devents(end).type, 'robot.cmd'))
18            switch(devents(end).value)
19                case 'left'
20                    fprintf('GO LEFT');
21                    sendDriveCommand('l', 'low');
22                case 'forward'
23                    fprintf('GO FORWARD');
24                    sendDriveCommand('f', 'low');
25                case 'right'
26                    fprintf('GO RIGHT');
27                    sendDriveCommand('r', 'low');
28            end;
29        end
30    end

```

```

31 end
32
33 closeHandle();
34 resetAll();

```

```

1 %% startSigProcBuffer.m
2 % buffer controlled execution of the different signal processing ...
  phases.
3 %
4 % Input events: (type,value)
5 % (startPhase.cmd,capfitting) — show capfitting
6 % (startPhase.cmd,calibrate) — start calibration phase ...
  processing (i.e. cat data)
7 % (startPhase.cmd,testing) — start test phase, i.e. on-line ...
  prediction generation
8 % (startPhase.cmd,exit) — stop everything
9 run ../utilities/initPaths.m;
10 configureIM;
11
12 if ( ~exist('capFile','var') ) capFile='1010'; end; ...
  %'cap_tmsi_mobita_num';
13 if ( ~isempty(strfind(capFile,'tmsi')) ) thresh=[.0 .1 .2 5]; ...
  badchThresh=1e-4; overridechnms=1;
14 else thresh=[.5 3]; ...
  badchThresh=.5; overridechnms=0;
15 end
16 datestr = datevec(now); datestr = ...
  sprintf('%02d%02d%02d',datestr(1)-2000,datestr(2:3));
17 dname='training_data';
18 cname='clsfr';
19 testname='testing_data';
20 if ( ~exist('verb','var') ) verb =2; end;
21 if ( ~exist('trlen_ms_ol','var') ) trlen_ms_ol=trlen_ms; end;
22 subject='test';
23
24 % startup the buffer etc. to wait for phase control events.
25 buffhost='localhost'; buffport=1972;
26 global ft_buff; ft_buff=struct('host',buffhost,'port',buffport);
27 % wait for the buffer to return valid header information
28 hdr=[];
29 while ( isempty(hdr) || ~isstruct(hdr) || (hdr.nchans==0) ) % ...
  wait for the buffer to contain valid data
30 try
31   hdr=buffer('get_hdr',[],buffhost,buffport);
32 catch
33   fprintf('Waiting for header\n');
34   hdr=[];
35 end;
36 pause(1);
37 end;
38
39 % main loop waiting for commands and then executing them
40 state=struct('pending',[],'nevents',[],'nsamples',[],'hdr',hdr);
41 phaseToRun=[]; clsSubj=[]; trainSubj=[];
42 while ( true )
43
44   if ( ~isempty(phaseToRun) ) state=[]; end
45   drawnow;
46
47   % wait for a phase control event
48   if ( verb>0 ) fprintf('Waiting for phase command\n'); end;

```

```

49 [data,devents,state]=buffer_waitData(buffhost, buffport, ...
    state, 'trlen_ms',0, 'exitSet',{{'startPhase.cmd' ...
        'subject'}}}, 'verb',verb,'timeOut_ms',5000);
50 if ( numel(devents)==0 )
51     continue;
52 elseif ( numel(devents)>1 )
53     % ensure events are processed in *temporal* order
54     [ans,eventorder]=sort([devents.sample],'ascend');
55     data=data(eventorder); devents=devents(eventorder);
56 end
57 if ( verb>0 ) fprintf('Got Event: %s\n',ev2str(devents)); end;
58
59 % extract the subject info
60 phaseToRun=[];
61 for di=1:numel(devents);
62     % extract the subject info
63     if ( strcmp(devents(di).type,'subject') )
64         subject=devents(di).value;
65         if ( verb>0 ) fprintf('Setting subject to : ...
            %s\n',subject); end;
66         continue;
67     else
68         phaseToRun=devents(di).value;
69         break;
70     end
71 end
72 if ( isempty(phaseToRun) ) continue; end;
73
74 fprintf('%d) Starting phase : ...
    %s\n',devents(di).sample,phaseToRun);
75
76 switch lower(phaseToRun);
77
78     %-----
79 case 'capfitting';
80     if ( verb>0 ) fprintf('Starting : %s\n',phaseToRun); ...
        ptime=getwTime(); end;
81     sendEvent(lower(phaseToRun),'start'); % mark start/end testing
82     capFitting('noiseThresholds',thresh,'badChThreshold',badchThresh, ...
        'verb',verb, 'showOffset',0, 'capFile',capFile, ...
        'overridechnms',overridechnms);
83     sendEvent(lower(phaseToRun),'end'); % mark start/end testing
84     if ( verb>0 ) fprintf('Finished : %s @ ...
        %5.3fs\n',phaseToRun,getwTime()-ptime); end;
85
86     %-----
87 case 'eegviewer';
88     if ( verb>0 ) fprintf('Starting : %s\n',phaseToRun); ...
        ptime=getwTime(); end;
89     sendEvent(lower(phaseToRun),'start'); % mark start/end testing
90     eegViewer(buffhost,buffport, 'capFile',capFile, ...
        'overridechnms',overridechnms);
91     sendEvent(lower(phaseToRun),'end'); % mark start/end testing
92     if ( verb>0 ) fprintf('Finished : %s @ ...
        %5.3fs\n',phaseToRun,getwTime()-ptime); end;
93
94     %-----
95 case {'calibrate','calibration'};
96     if ( verb>0 ) fprintf('Starting : %s\n',phaseToRun); ...
        ptime=getwTime(); end;
97     [traindata,traindevents,state]=buffer_waitData(buffhost, ...
        buffport, [], 'startSet',{{'stimulus.target'}}, ...

```

```

        'exitSet',{ 'stimulus.training' 'end'}, 'verb',verb, ...
        'trlen_ms',trlen_ms);
98  mi=matchEvents(traindevents,'stimulus.training','end'); ...
    traindevents(mi)=[];traindata(mi)=[];%remove exit event
99  fprintf('Saving %d epochs to : ...
        %s\n',numel(traindevents),[dname '_' subject '_' datestr]);
100 save([dname '_' subject '_' ...
        datestr],'traindata','traindevents');
101 trainSubj=subject;
102 sendEvent(lower(phaseToRun),'end'); % mark start/end testing
103 if ( verb>0 ) fprintf('Finished : %s @ ...
        %5.3fs\n',phaseToRun,getTime()-ptime); end;
104
105 %-----
106 case {'train','training'};
107 %try
108 if ( verb>0 ) fprintf('Starting : %s\n',phaseToRun); ...
    ptime=getwTime(); end;
109 if ( ~isequal(trainSubj,subject) || ~...
    exist('traindata','var') )
110 fprintf('Loading training data from : %s\n',[dname '_' ...
    subject '_' datestr]);
111 load([dname '_' subject '_' datestr]);
112 trainSubj=subject;
113 end;
114 if ( verb>0 ) fprintf('%d epochs\n',numel(traindevents)); end;
115 sendEvent(lower(phaseToRun),'start'); % mark start/end testing
116 clsfr=buffer_train_ersp_clsfr(traindata, traindevents, ...
    state.hdr, 'spatialfilter','ssep2', 'freqband',{[13 17 ...
    23]}, 'badchrm',1, 'badtrrm',1, 'objFn','lr_cg', ...
    'compKernel',0, 'dim',3, 'capFile',capFile, ...
    'overridechnms',overridechnms, 'visualize',2, ...
    'width_ms',1000);
117 %clsfr=buffer_train_ersp_clsfr(traindata, ...
    traindevents,state.hdr, 'spatialfilter','slap', ...
    'freqband',{[13 15 31 33]}, 'badchrm',1, 'badtrrm',1, ...
    'objFn','lr_cg', 'compKernel',0, 'dim',3, ...
    'capFile',capFile, 'overridechnms',overridechnms, ...
    'visualize',2, 'width_ms',1000);
118 clsSubj=subject;
119 fprintf('Saving classifier to : %s\n',[cname '_' subject ...
    '_' datestr]);
120 save([cname '_' subject '_' datestr'],'-struct','clsfr');
121 if ( verb>0 ) fprintf('Finished : %s @ ...
    %5.3fs\n',phaseToRun,getTime()-ptime); end;
122 %catch
123 % fprintf('Error in train classifier!');
124 %end
125 sendEvent(lower(phaseToRun),'end'); % mark start/end testing
126
127 %-----
128 case {'test','testing'};
129 if ( verb>0 ) fprintf('Starting : %s\n',phaseToRun); ...
    ptime=getwTime(); end;
130 if ( ~isequal(clsSubj,subject) || ~exist('clsfr','var') )
131 clsfrfile = [cname '_' subject '_' datestr];
132 if ( ~exist([clsfrfile '.mat'],'file') ) clsfrfile=[cname ...
    '_' subject]; end;
133 if(verb>0)fprintf('Loading classifier from file : ...
    %s\n',clsfrfile);end;
134 clsfr=load(clsfrfile);
135 clsSubj = subject;

```

```

136     end;
137     sendEvent(lower(phaseToRun),'start'); % mark start/end testing
138     imOnlineFeedbackSignals(clsfr, 'buffhost',buffhost, ...
        'buffport',buffport, 'hdr',hdr, 'trlen_ms',trlen_ms_ol)
139     sendEvent(lower(phaseToRun),'end');
140     if ( verb>0 ) fprintf('Finished : %s @ ...
        %5.3fs\n',phaseToRun,getTime()-ptime); end;
141
142     case 'exit';
143         break;
144
145     otherwise;
146         warning(sprintf('Unrecognised experiment phase ignored! : ...
            %s',phaseToRun));
147
148     end
149 end
150
151 %uiwait(msgbox({'Thank you for participating in our ...
        experiment.'},'Thanks','modal'),10);

```

9.3 Data processing and plotting files

```
1 function [ ] = GJ_prepData( subject, degrees, order )
2 %%[ ] = GJ_prepData( subject, degrees, order )
3 %
4 % load raw data, preprocess and save preprocessed data of ...
5 % specified order
6 % subject = 'GJ' | 'JT' | 'HV'
7 % degrees = (16:-2:6)
8 % order = 'ord' | 'rnd' (tests done in order or random)
9
10 for i = 1:numel(degrees)
11     fileName = ['rawData_' subject '_' num2str(degrees(i)) '_' ...
12               order];
13     load(fileName);
14
15     if(degrees(i) == 16) %closest measurement, train classifier
16
17         % preprocess plus train classifier on 16deg data with ...
18         % ssep filter
19         [clsfr,res,X,Y]=buffer_train_ersp_clsfr(traindata, ...
20         traindevents, ...
21         [], 'spatialfilter', 'ssep', 'freqband', {[13 17 ...
22         23]}, 'badchrm', 1, 'badtrrm', 1, 'objFn', 'lr_cg', ...
23         'compKernel', 0, 'dim', 3, 'capFile', 'emotiv1', ...
24         'overridechnms', 1, 'visualize', 1, 'fs', 128, ...
25         'width_ms', 1000);
26     % save classifier
27     save(['clsfr16_' subject '_' order] , 'clsfr');
28
29     % save preprocessed data
30     saveName = ['prepData_' subject '_' num2str(degrees(i)) ...
31               '_' order '_clsfr16'];
32     save(saveName, 'X', 'Y');
33
34     else % other measurements, apply clsfr16 on data to preprocess
35
36         % use classifier of first test (16 degrees) to apply to ...
37         % other tests
38         load('clsfr16');
39         [f, fraw, p, X]=buffer_apply_ersp_clsfr(traindata, clsfr, 1);
40
41         % save preprocessed data including events
42         saveName = ['prepData_' subject '_' num2str(degrees(i)) ...
43                   '_' order '_clsfr16'];
44         save(saveName, 'X', 'traindevents');
45     end
46 end
```

```
1 function [ampXstim13 ampXstim17 ampXstim23 ampXstimX] = ...
2     GJ_calcAmp(subject, degrees, order, isClsfr)
3 %%[ampXstim13 ampXstim17 ampXstim23 ampXstimX] = ...
4     GJ_calcAmp(subject, degrees, order, isClsfr)
5 %
6 % check average amp (over all trials) of 13/17/23Hz
7 % for each angle measurement (16-6deg)
8 % subject = 'GJ' | 'JT' | 'HV'
9 % degrees = (16:-2:6)
10 % order = 'ord' | 'ran'
```

```

9 % isClsfr = 0 | 1 (use clsfr of 16deg angle)
10 % determines which preproc data to open/use
11 %
12 % ampXstim13 = amps of 13 Hz trials
13 % ampXstim17 = amps of 17 Hz trials
14 % ampXstim23 = amps of 23 Hz trials
15 % ampXstimX = amps of each own trial
16
17 % channel(s) of interest
18 %channel=7; %01 (for slap filter)
19 %channel=8; %02 (for slap filter)
20 channel=1; %first 'virtual channel (for ssep filter)
21
22 numSF = 2; % number of spatial filters (max for ssep filter)
23 ampXstim13 = zeros(numSF,3,numel(degrees));
24 ampXstim17 = zeros(numSF,3,numel(degrees));
25 ampXstim23 = zeros(numSF,3,numel(degrees));
26 ampXstimX = zeros(numSF,3,numel(degrees));
27
28 %load each data degree measurement and calc avg amps
29 for i = 1:numel(degrees) % number of degrees
30
31     if (isClsfr && degrees(i)≠16) % anything but the 16th ...
32         degree, no Y is saved so used trainevents
33         fileName = ['prepData_' subject '_' num2str(degrees(i)) ...
34                     '_' order '_clsfr16'];
35         load(fileName);
36         % indices of trials for each freq
37         idx13 = find([trainevents.value] == 1);
38         idx17 = find([trainevents.value] == 2);
39         idx23 = find([trainevents.value] == 3);
40     elseif(isClsfr) % use data of clsfr16
41         fileName = ['prepData_' subject '_' num2str(degrees(i)) ...
42                     '_' order '_clsfr16'];
43         load(fileName);
44         % indices of trials for each freq
45         idx13 = find(Y==1);
46         idx17 = find(Y==2);
47         idx23 = find(Y==3);
48     else
49         fileName = ['prepData_' subject '_' num2str(degrees(i)) ...
50                     '_' order ];
51         load(fileName);
52         % indices of trials for each freq
53         idx13 = find(Y==1);
54         idx17 = find(Y==2);
55         idx23 = find(Y==3);
56     end
57
58     % first spatial filter, 13 hz trials
59     ampXstim13(1,i,1) = mean(X(channel,1,idx13)); % avg amp of ...
60     % 13 for 13hz trials
61     ampXstim13(2,i,1) = mean(X(channel,2,idx13)); % avg amp of ...
62     % 17 for 13hz trials
63     ampXstim13(3,i,1) = mean(X(channel,3,idx13)); % avg amp of ...
64     % 23 for 13hz trials
65
66     % second spatial filter, 13 hz trials
67     ampXstim13(1,i,2) = mean(X(channel+1,1,idx13)); % avg amp of ...
68     % 13 for 13hz trials
69     ampXstim13(2,i,2) = mean(X(channel+1,2,idx13)); % avg amp of ...
70     % 17 for 13hz trials

```

```

62     ampXstim13(3,i,2) = mean(X(channel+1,3,idx13)); % avg amp of ...
        23 for 13hz trials
63
64     % first spatial filter, 17 hz trials
65     ampXstim17(1,i,1) = mean(X(channel,1,idx17)); % avg amp of ...
        13 for 17hz trials
66     ampXstim17(2,i,1) = mean(X(channel,2,idx17)); % avg amp of ...
        17 for 17hz trials
67     ampXstim17(3,i,1) = mean(X(channel,3,idx17)); % avg amp of ...
        23 for 17hz trials
68
69     % second spatial filter, 17 hz trials
70     ampXstim17(1,i,2) = mean(X(channel+1,1,idx17)); % avg amp of ...
        13 for 17hz trials
71     ampXstim17(2,i,2) = mean(X(channel+1,2,idx17)); % avg amp of ...
        17 for 17hz trials
72     ampXstim17(3,i,2) = mean(X(channel+1,3,idx17)); % avg amp of ...
        23 for 17hz trials
73
74     % first spatial filter, 23 hz trials
75     ampXstim23(1,i,1) = mean(X(channel,1,idx23)); % avg amp of ...
        13 for 23hz trials
76     ampXstim23(2,i,1) = mean(X(channel,2,idx23)); % avg amp of ...
        17 for 23hz trials
77     ampXstim23(3,i,1) = mean(X(channel,3,idx23)); % avg amp of ...
        23 for 23hz trials
78
79     % second spatial filter, 23 hz trials
80     ampXstim23(1,i,2) = mean(X(channel+1,1,idx23)); % avg amp of ...
        13 for 23hz trials
81     ampXstim23(2,i,2) = mean(X(channel+1,2,idx23)); % avg amp of ...
        17 for 23hz trials
82     ampXstim23(3,i,2) = mean(X(channel+1,3,idx23)); % avg amp of ...
        23 for 23hz trials
83
84     % first spatial filter, each own trials
85     ampXstimX(1,i,1) = mean(X(channel,1,idx13)); % avg amp of 13 ...
        for 13hz trials
86     ampXstimX(2,i,1) = mean(X(channel,2,idx17)); % avg amp of 17 ...
        for 17hz trials
87     ampXstimX(3,i,1) = mean(X(channel,3,idx23)); % avg amp of 23 ...
        for 23hz trials
88
89     % second spatial filter, each own trials
90     ampXstimX(1,i,2) = mean(X(channel+1,1,idx13)); % avg amp of ...
        13 for 13hz trials
91     ampXstimX(2,i,2) = mean(X(channel+1,2,idx17)); % avg amp of ...
        17 for 17hz trials
92     ampXstimX(3,i,2) = mean(X(channel+1,3,idx23)); % avg amp of ...
        23 for 23hz trials
93     end

```

```

1 function [combAmpStim13,combAmpStim17,combAmpStim23,combAmpStimX ...
    ] = GJ_combnOrdRan( subject,degrees,isClsfr )
2 %%[combAmpStim13,combAmpStim17,combAmpStim23,combAmpStimX ] = ...
    GJ_combnOrdRan( subject,degrees,isClsfr )
3 %
4 %function to combine data (in this case average amplitudes) of ...
    both ordered
5 %and random experiment setup from ssvep measurements.
6 %subject = 'GJ' | 'JT' | 'HV'

```

```

7 %degrees = 16:-2:6
8 %isClsfr = 1 | 0 (use classifier of 16th degree for all others)
9 %
10 %combAmpStim13 = combined amps of 13hz stim
11 %combAmpStim17 = combined amps of 17hz stim
12 %combAmpStim23 = combined amps of 23hz stim
13 %combAmpStim X= combined amps of each own stim
14
15 % calculate amplitudes of both ordered and random experiment
16 [ampXstim13_o ampXstim17_o ampXstim23_o ampXstimX_o] = ...
    GJ_calcAmp( subject,degrees,'ord',isClsfr);
17 [ampXstim13_r ampXstim17_r ampXstim23_r ampXstimX_r] = ...
    GJ_calcAmp( subject,degrees,'ran',isClsfr);
18
19 % average of both amplitudes (ordered/random)
20 combAmpStim13 = (ampXstim13_o + ampXstim13_r)/2;
21 combAmpStim17 = (ampXstim17_o + ampXstim17_r)/2;
22 combAmpStim23 = (ampXstim23_o + ampXstim23_r)/2;
23 combAmpStimX = (ampXstimX_o + ampXstimX_r)/2;
24
25 % create saveName depending on using clsfr16
26 if(isClsfr)
27     saveName = ['combData_' subject '_clsfr16'];
28 else
29     saveName = ['combData_' subject];
30 end
31
32 % save combined data of subject
33 save(saveName,'combAmpStim13','combAmpStim17','combAmpStim23', ...
    'combAmpStimX');
34
35 end

```

```

1 %%GJ_combnPartc.m
2 %
3 % script to combine participant's data
4 % loads data of each participant and average over all three
5 % saves to combData_allPartc_clsfr16.mat
6
7 % load data
8 A = load('combData_GJ_clsfr16');
9 B = load('combData_JT_clsfr16');
10 C = load('combData_HV_clsfr16');
11
12 % combine/average data
13 comb.ampXstim13 = ...
    (A.combAmpStim13+B.combAmpStim13+C.combAmpStim13)/3;
14 comb.ampXstim17 = ...
    (A.combAmpStim17+B.combAmpStim17+C.combAmpStim17)/3;
15 comb.ampXstim23 = ...
    (A.combAmpStim23+B.combAmpStim23+C.combAmpStim23)/3;
16 comb.ampXstimX = (A.combAmpStimX+B.combAmpStimX+C.combAmpStimX)/3;
17
18 % save combined data
19 save('combData_allPartc_clsfr16','-struct','comb');

```

```

1 function [] = GJ_mergeAndprepData(subject,degrees)
2 %%[] = GJ_mergeAndprepData(subject,degrees)
3 %

```

```

4 % load raw data, merge, preprocess and save preprocessed data
5 % subject = 'GJ' | 'JT' | 'HV'
6 % degrees = (16:-2:6)
7
8 for i = 1:numel(degrees)
9
10     % load data of ordered tests
11     fileName = ['rawData_' subject '_' num2str(degrees(i)) '_ord'];
12     load(fileName);
13
14     % temporary name for ordered data
15     a1=traindata;
16     b1=traindevents;
17
18     % load data of random tests
19     fileName2 = ['rawData_' subject '_' num2str(degrees(i)) '_ran'];
20     load(fileName2);
21
22     % merge random and ordered trial data
23     traind = [traindata; a1];
24     traindev = [traindevents; b1];
25
26     % save merged raw data (traind = data, traindev = events)
27     saveName=['mergedRawData_' subject '_' num2str(degrees(i))];
28     save(saveName,'traind','traindev');
29
30     % load merged data
31     load(saveName);
32
33     if(degrees(i) == 16) %closest measurement, train classifier
34
35         % preprocess plus train classifier on 16deg data with ...
36         % ssep filter
37         [clsfr,res,X,Y]=buffer_train_ersp_clsfr(traind, ...
38         traindev, [], 'spatialfilter','ssep','freqband',{[13 ...
39         17 23]},'badchrm',1,'badtrrm',1,'objFn','lr_cg', ...
40         'compKernel',0,'dim',3,'capFile','emotiv1', ...
41         'overridechnms',1,'visualize',2,'fs',128, ...
42         'width_ms',1000);
43     % save classifier
44     save(['clsfr16_' subject '_merged'] , 'clsfr');
45
46     % save preprocessed data
47     saveName = ['mergedPrepData_' subject '_' ...
48     num2str(degrees(i)) '_clsfr16'];
49     save(saveName,'X','Y');
50 else % other measurements, apply clsfr16 on data to preprocess
51
52     % use classifier of first test (16 degrees) to apply to ...
53     % other tests
54     load(['clsfr16_' subject '_merged']);
55     [f,fraw,p,X]=buffer_apply_ersp_clsfr(traind,clsfr,2);
56
57     % save preprocessed data including events and raw ...
58     % decision values
59     saveName = ['mergedPrepData_' subject '_' ...
60     num2str(degrees(i)) '_clsfr16'];
61     save(saveName,'X','traindev','p');
62 end
63 end

```

```

1 function [ hitRate13 hitRate17 hitRate23 hitRate] = ...
    GJ_calcHitrate( subject,degrees )
2 %%[ hitRate13 hitRate17 hitRate23 hitRate] = GJ_calcHitrate( ...
    subject,degrees )
3 %
4 % calculate true positive rate of each class and total true ...
    positive rate
5 % subject = 'GJ'| 'JT' | 'HV'
6 % degrees = (16:-2:6)
7 %
8 % hitRate = total true positive rate
9 % hitRate13 = true positive rate of 13 Hz trials
10 % hitRate17 = true positive rate of 17 Hz trials
11 % hitRate23 = true positive rate of 23 Hz trials
12
13 %calculated hitRates (first is 16 degrees)
14 hitRate = [];
15 hitRate13 = [];
16 hitRate17 = [];
17 hitRate23 = [];
18
19 for ii = 1:numel(degrees)
20     truePos13 = 0;
21     truePos17 = 0;
22     truePos23 = 0;
23
24     if(degrees(ii)<16) % used clsfr16 so only p available
25         loadName=['mergedPrepData_' subject '_' num2str(degrees(ii)) ...
            '_clsfr16'];
26         load(loadName);
27
28         %get max decision value = classified class
29         [val j] = max(p,[],2);
30
31         %get real labels from data
32         labels=[traindev.value];
33
34         for i=1:numel(j)
35             if(j(i) == 1 && j(i)==labels(i)) %class=1 + pred=correct
36                 truePos13=truePos13+1;
37             elseif(j(i) == 2 && j(i)==labels(i)) %class=2 + pred=correct
38                 truePos17=truePos17+1;
39             elseif( (j(i) == 3) && (j(i)==labels(i)) ) %class=3 + ...
                pred=correct
40                 truePos23=truePos23+1;
41         end
42     end
43
44     % total correct classifications / total classifications = ...
        true positive
45     hitRate(ii) = sum(j==labels(:))/numel(labels); %precision ...
        (tp/tp+fp)
46     hitRate13(ii) = truePos13/sum(labels==1);
47     hitRate17(ii) = truePos17/sum(labels==2);
48     hitRate23(ii) = truePos23/sum(labels==3);
49
50 else %clsfr16 is already applied to deg=16, get hitrate from ...
    apply instead of train
51     loadName=['mergedPrepData_' subject '_' num2str(degrees(ii)) ...
        '_clsfr16_applied'];
52     load(loadName);
53

```

```

54 %get max decision value = classified class
55 [val j] = max(p,[],2);
56
57 %get real labels from data
58 labels=[traindev.value];
59
60 for i=1:numel(j)
61     if(j(i) == 1 && j(i)==labels(i)) %class=1 + pred=correct
62         truePos13=truePos13+1;
63     elseif(j(i) == 2 && j(i)==labels(i)) %class=2 + pred=correct
64         truePos17=truePos17+1;
65     elseif( (j(i) == 3) && (j(i)==labels(i)) ) %class=3 + ...
66         pred=correct
67         truePos23=truePos23+1;
68     end
69 end
70
71 % total correct classifications / total classifications = ...
72 true positive
73 hitRate(ii) = sum(j==labels(:))/numel(labels); %precision ...
74 (tp/tp+fp)
75 hitRate13(ii) = truePos13/sum(j==1);
76 hitRate17(ii) = truePos17/sum(j==2);
77 hitRate23(ii) = truePos23/sum(j==3);
78
79 end
80 end

```

```

1 function [ ] = GJ_plotData( subject,degrees,order,isClsfr,sndSF )
2 %%[ ] = GJ_plotData( subject,degrees,order,isClsfr,sndSF )
3 %
4 % plotting average (of all trials) amps of 13/17/23 Hz, with ...
5 % option to plot
6 %
7 % results of 16th degree classifier and 2nd spatial filter
8 %
9 % subject = 'GJ' | 'JT' | 'HV'
10 % degrees = (16:-2:6)
11 % order = 'ord' | 'ran'
12 % isClsfr = 0 | 1 (use clsfr of 16deg angle)
13 % sndSF = 0 | 1 (plot 2nd spatial filter results)
14
15 if(strcmp(order,'comb'))
16     % combine amplitudes of both ordered and random measurement
17     [ampXstim13,ampXstim17,ampXstim23,ampXstimX] = ...
18     GJ_combnOrdRan( subject,degrees,isClsfr);
19 elseif(strcmp(order,'allp'))
20     load('combData_allPartc_clsfr16');
21 else % calculate amplitudes of specified order and classifier
22     [ampXstim13 ampXstim17 ampXstim23 ampXstimX] = ...
23     GJ_calcAmp(subject, degrees,order,isClsfr);
24 end
25
26 % labels and titles for plots
27 labels = ['13Hz'; '17Hz'; '23Hz'];
28 titels = {'Amps for 13Hz stim'; 'Amps for 17Hz stim'; 'Amps for ...
29           23Hz stim'; 'Amps for each stimulus'};
30
31 % new figure
32 figure;
33 hold all;

```

```

29
30 % if 2nd spatial filter = 0, current SF is 1
31 SF=1;
32
33 %% first spatial filter
34 subplot(2,2,1)
35 plot(degrees,ampXstim13(1,:,1),'o-',degrees,ampXstim13(2,:,1), ...
      'o-', degrees,ampXstim13(3,:,1),'o-');
36 title(titels{1});
37 legend(labels);
38 xlabel('Degrees');
39 ylabel('Amplitude uV');
40
41 subplot(2,2,2)
42 plot(degrees,ampXstim17(1,:,1),'o-',degrees,ampXstim17(2,:,1), ...
      'o-', degrees,ampXstim17(3,:,1),'o-');
43 title(titels{2});
44 legend(labels);
45 xlabel('Degrees');
46 ylabel('Amplitude uV');
47
48 subplot(2,2,3)
49 plot(degrees,ampXstim23(1,:,1),'o-',degrees,ampXstim23(2,:,1), ...
      'o-', degrees,ampXstim23(3,:,1),'o-');
50 title(titels{3});
51 legend(labels);
52 xlabel('Degrees');
53 ylabel('Amplitude uV');
54
55 subplot(2,2,4)
56 plot(degrees,ampXstimX(1,:,1),'o-',degrees,ampXstimX(2,:,1), ...
      'o-', degrees,ampXstimX(3,:,1),'o-');
57 title(titels{4});
58 legend(labels,'Location','SouthOutside');
59 xlabel('Degrees');
60 ylabel('Amplitude uV');
61
62 suptitle(['spatfilt: ' num2str(SF) ', subj: ' subject ', order: ...
          ' order ', clsfr16: ' int2str(isClsfr)']);
63
64 %% second spatial filter
65
66 if(sndSF)
67     % if 2nd spatial filter = 1, current SF is 2
68     SF=SF+1;
69
70     % new 2nd figure
71     figure;
72     hold all;
73
74     [ampXstim13 ampXstim17 ampXstim23 ampXstimX] = ...
        GJ_calcAmp(subject, degrees,order,isClsfr);
75     labels = ['13Hz'; '17Hz'; '23Hz'];
76     titels = {'Amps for 13Hz stim'; 'Amps for 17Hz stim'; 'Amps ...
              for 23Hz stim'; 'Amps for each stimulus'};
77
78     subplot(2,2,1)
79     plot(degrees,ampXstim13(1,:,2),'o-',degrees,ampXstim13(2,:,2) ...
          'o-', degrees,ampXstim13(3,:,2),'o-');
80     title(titels{1});
81     legend(labels);
82     xlabel('Degrees');

```

```

83     ylabel('Amplitude uV');
84
85     subplot(2,2,2)
86     plot(degrees,ampXstim17(1,:,2),'o-',degrees,ampXstim17(2,:,2) ...
87           , 'o-', degrees,ampXstim17(3,:,2),'o-');
88     title(titels(2));
89     legend(labels);
90     xlabel('Degrees');
91     ylabel('Amplitude uV');
92
93     subplot(2,2,3)
94     plot(degrees,ampXstim23(1,:,2),'o-',degrees,ampXstim23(2,:,2) ...
95           , 'o-', degrees,ampXstim23(3,:,2),'o-');
96     title(titels(3));
97     legend(labels);
98     xlabel('Degrees');
99     ylabel('Amplitude uV');
100
101    subplot(2,2,4)
102    plot(degrees,ampXstimX(1,:,2),'o-',degrees,ampXstimX(2,:,2) ...
103          , 'o-', degrees,ampXstimX(3,:,2),'o-');
104    title(titels(4));
105    legend(labels);
106    xlabel('Degrees');
107    ylabel('Amplitude uV');
108
109    suptitle(['spatfilt: ' num2str(SF) ', subj: ' subject ', ...
110             order: ' order ', clsfr16: ' int2str(isClsfr)']);
111 end
112 end

```

```

1  function [ ] = GJ_plot( subject,degrees,order)
2  %%[ ] = GJ_plot( subject,degrees,order)
3  %
4  % plotting subject's average (of all trials) amps of 13/17/23 Hz,
5  % subject = 'GJ' | 'JT' | 'HV' | 'allp'
6  % degrees = (16:-2:6)
7  % order = 'comb' | 'allp'
8
9  % Load data
10 if(strcmp(subject,'allp'))% load combined data of all subjects
11     load('combData_allPartc_clsfr16');
12     combAmpStim13 = ampXstim13;
13     combAmpStim17 = ampXstim17;
14     combAmpStim23 = ampXstim23;
15     combAmpStimX = ampXstimX;
16     subject='allPartc';
17 elseif(strcmp(order,'comb'))% load combined data of subject
18     load(['combData_' subject '_clsfr16']);
19 else
20     disp('error, no order defined');
21 end
22
23 % legend & titles
24 labels = ['13Hz'; '17Hz'; '23Hz'];
25 titels = {'Amplitude vs TP-rate for 13Hz trials'; 'Amplitude vs ...
26           TP-rate for 17Hz trials'; 'Amplitude vs TP-rate for 23Hz ...
27           trials'; 'Amplitude vs TP-rate for each own trials'};

```

```

28 amps13 = combAmpStim13(:, :, 1);
29 amps17 = combAmpStim17(:, :, 1);
30 amps23 = combAmpStim23(:, :, 1);
31 ampsX = combAmpStimX(:, :, 1);
32
33 % calculate hitrate of each stimulus
34 [hr13 hr17 hr23 hrX] = GJ_calcHitrate(subject, degrees);
35
36 % new figure, add suptitle above subplots
37 figure;
38 hold on;
39 suptitle(['subj: ' subject ', spatfilt: 1st SSEP']);
40
41 %% 13hz stim
42 subplot(2, 2, 1)
43 [AX, H1, H2]=plotyy(degrees, amps13, degrees, hr13, 'plot', 'bar');
44 set(get(AX(1), 'Ylabel'), 'String', 'Amplitude (uV)')
45 set(get(AX(2), 'Ylabel'), 'String', 'True positive rate (%)')
46 ylim([0 0.4])
47 ymax = get(AX(1), 'YLim');
48 set(AX(1), 'YTick', linspace(0, ymax(2), 11));
49 set(AX(2), 'YTick', (0:0.1:1));
50 set(H1, 'LineStyle', '-', 'Marker', 'o');
51 set(H2, 'FaceColor', 'none', 'BarWidth', 0.5);
52 legend(labels);
53 xlabel('Visual angle (degrees)');
54 title(titels{1});
55
56 %% 17hz stim
57 subplot(2, 2, 2)
58 [AX, H1, H2]=plotyy(degrees, amps17, degrees, hr17, 'plot', 'bar');
59 set(get(AX(1), 'Ylabel'), 'String', 'Amplitude (uV)')
60 set(get(AX(2), 'Ylabel'), 'String', 'True positive rate (%)')
61 ylim([0 0.4])
62 ymax = get(AX(1), 'YLim');
63 set(AX(1), 'YTick', linspace(0, ymax(2), 11));
64 set(AX(2), 'YTick', (0:0.1:1));
65 set(H1, 'LineStyle', '-', 'Marker', 'o');
66 set(H2, 'FaceColor', 'none', 'BarWidth', 0.5);
67 legend(labels);
68 xlabel('Visual angle (degrees)');
69 title(titels{2});
70
71
72 %% 23hz stim
73 subplot(2, 2, 3)
74 [AX, H1, H2]=plotyy(degrees, amps23, degrees, hr23, 'plot', 'bar');
75 set(get(AX(1), 'Ylabel'), 'String', 'Amplitude (uV)')
76 set(get(AX(2), 'Ylabel'), 'String', 'True positive rate (%)')
77 ylim([0 0.4])
78 ymax = get(AX(1), 'YLim');
79 set(AX(1), 'YTick', linspace(0, ymax(2), 11));
80 set(AX(2), 'YTick', (0:0.1:1));
81 set(H1, 'LineStyle', '-', 'Marker', 'o');
82 set(H2, 'FaceColor', 'none', 'BarWidth', 0.5);
83 legend(labels);
84 xlabel('Visual angle (degrees)');
85 title(titels{3});
86
87 %% each stim
88 subplot(2, 2, 4)
89 hold on;

```

```

90 [AX,H1,H2]=plotyy(degrees,ampsX,degrees,hrX,'plot','bar');
91 set(get(AX(1),'Ylabel'),'String','Amplitude (uV)')
92 set(get(AX(2),'Ylabel'),'String','True positive rate (%)')
93 ylim([0 0.4])
94 ymax = get(AX(1),'YLim');
95 set(AX(1),'YTick',linspace(0,ymax(2),11));
96 set(AX(2),'YTick',(0:0.1:1));
97 set(H1,'LineStyle','-','Marker','o');
98 set(H2,'FaceColor','none','BarWidth',0.5);
99 legend(labels);
100 xlabel('Visual angle (degrees)');
101 title(titels{4});
102
103 end

```

```

1 %%script to calculate horizontal distance from a subject with ...
  certain height
2
3 distBtwnLED = 40 % dist between LEDs (left/right)
4 distBtwn = 20; % dist between LEDs (middle vs left/right)
5 degrees = (2:2:22); % degrees apart of LEDs (steps of 2)
6 sizeStim = 4; % size of stimulus in cm
7 eyeHeight = 119.0 % height of subject's eyes
8
9 % half all for calculations
10 overstaande = distBtwnLED/2;
11 overstaandel = distBtwn/2;
12 halfdegrees = degrees/2;
13 halfSizeStim = sizeStim/2;
14
15 % geometry
16 radians = halfdegrees * pi/180; % convert to radians
17 reqDist = overstaande ./ tan(radians); % diagonal distance ...
  needed for angle
18 reqDist1 = overstaandel ./ tan(radians);
19
20 % init arrays
21 horzDist = zeros(numel(reqDist), 1);
22 angleStimLR = zeros(numel(reqDist),1);
23 angleStimML = zeros(numel(reqDist1),1);
24
25 % calculate horizontal distance for predefined angles
26 for i = 1:numel(reqDist)
27     % horizontal distance calc (a^2+b^2=c^2)
28     if ((reqDist(i)^2) > eyeHeight^2)
29         horzDist(i) = round(sqrt(reqDist(i)^2 - eyeHeight^2));
30         % angle of stim = tan(o/a) *180/pi * 2 (2x angle)
31         angleStimLR(i) = ((atan(halfSizeStim / reqDist(i))) * ...
32             180/pi) * 2;
33     else
34         horzDist(i) = NaN;
35         angleStimLR(i) = NaN;
36     end
37
38     if ((reqDist(i)^2) > eyeHeight^2)
39         %horzDist(i) = round(sqrt(reqDist1(i)^2 - eyeHeight^2));
40         % angle of stim = tan(o/a) *180/pi * 2 (2x angle)
41         angleStimML(i) = ((atan(halfSizeStim / reqDist1(i))) * ...
42             180/pi) * 2;
43     else
44         %horzDist(i) = NaN;

```

```
43         angleStimML(i) = NaN;
44     end
45 end
46
47 % show horizontal distances with each angle
48 degrWithDistCM = [degrees(:) horzDist(:)]
49 % show angle of stimulus for each distance
50 angleStimLR = angleStimLR(:) % between left and right
51 angleStimML = angleStimML(:) % between middle and left (or right)
```