SSVEP Evaluation for Robot Control

B. Specht, D. Stichling, A. Farshad, S. Martinez
Technical University of Munich
Arcisstrasse 21, 80333 Munich, Germany

Abstract

This is the project report for the course NeuroEngineering Implants, Interfaces and Algorithms. Methods and steps to use SSVEP-evaluation for robot control are discussed and introduced here. Furthermore, experiment settings and results are shown in this project report.

Introduction

Steady state visually evoked potentials (SSVEP) are signals measured after a visual stimulation. When the retina of an eye is excited by a visual stimulus ranging between 3.5 Hz and 75 Hz, one is able to measure a brain generated electrical potential at quite similar frequency called SSVEP. This effect can be used to interface the human brain and allow to control i.e. a robot with the evaluation of the SSVEP only. For this, one has to center his sight to an i.e. blinking object with mentioned frequency. With classification of the frequency of a certain object, extracted from recorded EEG data, a control pattern for looking on certain frequencies can be created and one can enable a certain control task by only looking at one certain blinking object. In this project, we want to implement the recognition of this SSVEP with certain frequency to enable the control of a robot and design an experiment to evaluate the functionality of the implementation. We want to compare different classification algorithms for this purpose and chose the best one fitting this project. People can use a brain-computer interface (BCI) to interact with their environment even if they have limited or no muscle control. A motivation of these devices is to get such in interaction limited people the capability to manipulate their environment with a robotic arm and give them independence in their daily life.

Experiment Designs

SSVEP-data can be acquired with techniques like electroencephalography (EEG), electrocorticography (ECoG), functional magnetic resonance imaging (fMRI) and near infrared spectroscopy (NIRS). EEG is one of the most common methods to access signals generated by the human brain and this is the method we will use, because it is more one of the inexpensive devices, non-invasive, and has good temporal resolution. However, the spatial resolution of an EEG is limited, as each channel electrode positioned roughly over a large amount of firing neurons. In addition, the signals generated by the human brain got distorted and filtered during the passage through the scalp and skull. Advanced methods of signal processing and machine learning approaches are usually used to reconstruct or retrieve useful information from these devices. To reach the goal, evaluate SSVEP and control a robot with a BCI, we decided to design two experiments. One experiment is used to record EEG-data to train several classifiers to detect SSVEP with certain frequencies. The second experiment records EEG-data for live classification with trained classifiers and controlling a robot with the recognized pattern.

Training Data Recording Experiment

To evoke SSVEP, a visual stimulus with a frequency between 3.5 Hz to 75 Hz needs to be presented to a subject. Data acquired from SSVEP-experiments in (1) suggests to use stimulation frequencies between 10 and 20 Hz, because these frequencies evoke the highest amplitude response and the highest SNR. This should provide the best data-quality to train a SSVEP-classifier. For this experiment we used four frequencies for stimuli: 13, 15, 17 and 19 Hz. With four frequencies to classify, one can encode five control actions for robot control. Five control actions, because a label like no SSVEP observed can be encoded too. To provide such stimuli, we decided to use a flickering animation on a screen. The screen should provide a picture refresh rate of 38 fps and more. So the flickering animation can be observed with the correct frequency without violating the Nyquist theorem. Therefore aliasing effects can be avoided. The animation is implemented with the python library pygame. Pygame is providing an API to render graphical user interfaces for gaming. The flickering object on the screen is realized by a checker board. Flickering is provided by inverting the color of the checker board. An important task is to verify the flickering frequency, because the animation of the object is not provided in real-time on a system without a real-time kernel. To minimize jitter caused by a systems process scheduler the animation is run in an own process with a frequency controller. The frequency controller observes the difference between set frequency

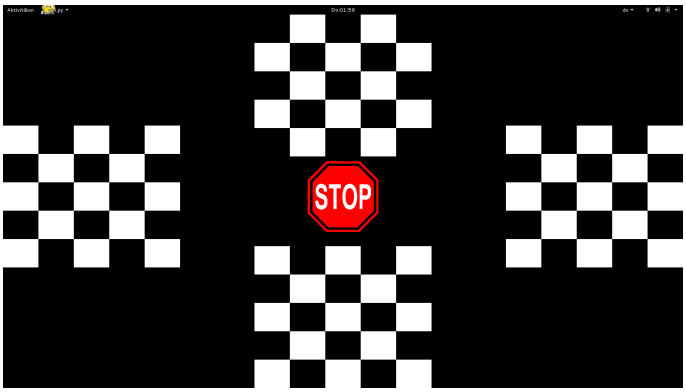

Fig. 1. Screenshot of the experiment 2 pygame-GUI. The four panels on the edges are the flickering checker board like surfaces. Arrangement and design of panels is adapted from (2). In the middle, a object is indicates the actual action of the robot. The current indicated action is here a stop command.

and observed frequency and regulates the waiting time between two animations. With this, the initialization error of the waiting time can be decreased and frequency errors caused by the system's workload changes minimized. The frequency controller is implemented with a PID-controller. Nevertheless, with a frequency controller a deviation of maximum 2 ms can be observed with a single animation on the test computer. Please consider, this deviation depends heavily on a computer system's workload and setup! For the actual data recording the electrodes O1, OZ and O2 on an EEG-cap were used. (1) suggested to use more electrodes and apply a certain algorithm to identify the electrode providing the best data for classification. However, for simplification we decided to use all data from the three electrodes next to the location where the visual cortex of the human brain would be. One trial of the experiment is the recording of EEG-data for 15 seconds while a stimulus of a certain frequency is provided. After the provided stimulus a 5 second long black screen is shown. The whole experiment for one frequency consists of 20 trials with 5 second long breaks. A break between two stimuli is needed, because long stimulation is quite uncomfortable and tiring.

Robot Control Experiment

The second experiment renders four checker boards on a screen, flickering with the frequencies of 13, 15, 17 and 19 Hz. In the middle of the screen an arrow points at the checker board blinking with the frequency that was recognized by a classifier. The robot is a robotic arm with 6 degrees of freedom. Movement of the robot is restricted to four directions in this experiment. In figure 1 one can see the GUI for this experiment.

A subject has to look at a specific panel to control the robot's movement. The direction of the movement is similar to the arrangement of the particular panel on the screen. For example, the robot will move left, if the test person is looking on the left panel. The task of the subject is to control the robot only by looking on one of the panels. The goal is to just play with the controls and test the quality of a live classification and recording of SSVEP-data.
The training data for the live classifier ensemble were altered between the data only recorded of the actual subject right before the second experiment and data from a general dataset of subjects recorded earlier with our provided first experiment.
In a second part the test person has to control the robot with a keyboard's arrow keys. With this procedure, a test person can evaluate the difference between the control with SSVEP-evaluation and control with a keyboard. After the experiment, the test person is asked to evaluate and compare this two kinds of robotic controls.

Experiment Procedure

Four subjects participated in all experiments. First, experiment 1 was carried out. From four subjects training data was collected for the SSVEP with frequencies 13, 15, 17 and 19 Hz. Experiment 2 was carried out with two subjects and live data was classified with classifiers trained on all subjects' data.
Due to the suspicion of varying experiment conditions, we decided to run a second procedure layout. The robot control experiment was carried out just after the recording experiment. So the variation of experiment conditions from other sessions might have gotten reduced. Only the just recorded dataset was later used for the live control experiment.

Recording

For the recording we used the "labstreaminglayer" library which receives the data from the electrodes through the amplifier and publishes it then to a python client. A gtec g.USBamp was used for this experiments with a sampling frequency of $256Hz$. Only the electrodes O1, OZ and O2 were used for this project. The training data for the classifiers was recorded continuously while marking the start of each trial with a time stamp. The first approach for recording and processing live data was appending a sample to a buffer and removing the oldest one. However, using locks when reading and modifying the structure turned out to be to slow. This led to the idea of using a single big buffer. As soon as the buffer was exhausted, writing started again at the beginning. Thus, old samples were overwritten. Figure 2 shows both approaches.

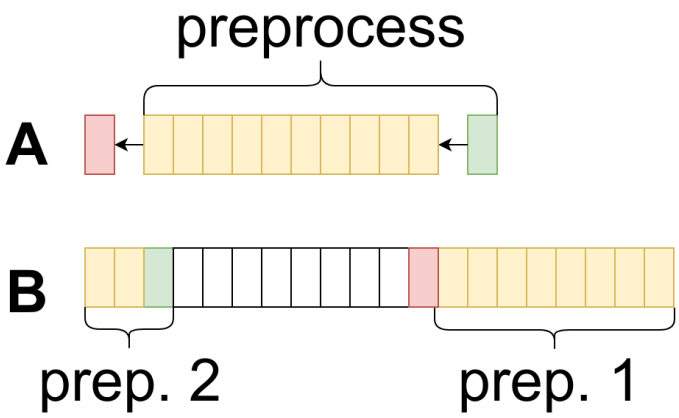

Fig. 2. ***A*** and ***B*** show both approaches of writing the live recorded data. In approach ***A*** every single time a new sample arrives, the whole buffer has to be reallocated in order to discard the oldest and add the newest sample. Additionally the data structure has to be locked, since it is read and modified from two independently running threads, which is quite slow. In ***B*** on the other hand the buffer is allocated once. A new incoming sample will be written to the current index. The writing will start again from the beginning, as soon as the index reaches the end.

Preprocessing of Data

Since the frequency the subject is being simulated with will also appear in the EEG it is useful to switch from time domain to frequency domain using for example the Fast-Fourier-Transform. The next step is cutting of the frequencies below 10 and above 20 Hz, because those parts of the spectrum do not include the stimulation frequencies. Removing the upper part makes a notch filter around 50 Hz to avoid the influence of the utility frequency unnecessary. One good way to represent the features is using the Power Density Spectrum (PDS) for the remaining spectrum. A stimulation frequency of 13 Hz will lead to a higher power density around the very same frequency. Figure 3 and figure 4 show the PDS plots for a stimulation frequency of 13 and 17 Hz.

Classification

We use different classifiers in this project. MLP or Multilayer Perceptron is a feed forward artificial neural network. LDA and QDA are methods used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. A performance evaluation of the used classifiers tested with cross validation is shown in figure 6.
PCA or Principal Component Analysis is another statistical procedure. It makes orthogonal transformations on the training data that allows us to project it into a smaller set, in order to minimize the computational cost in every reading. Then, it makes orthogonal transformations to compare the principal

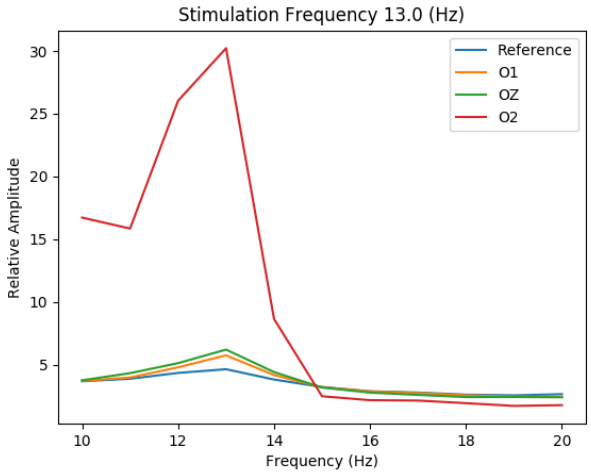

Fig. 3. When the subject is stimulated with a frequency of 13 Hz the Power Density Spectrum of the EEG signal will show peaks around the very same frequency.

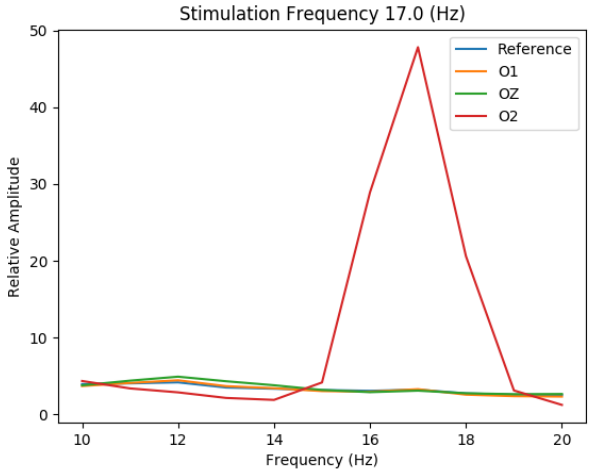

Fig. 4. When the subject is stimulated with a frequency of 17 Hz the Power Density Spectrum of the EEG signal will show peaks around the very same frequency.

components of the received test data, with the ones from each class by taking the highest likelihood. The following steps were applied to the training data in order to achieve results.

- Organize the data of each sample in an one-dimensional array. Since each sample has 4 readings (1 per electrode) of 11 values corresponding to the amplitude of the PDS in the frequencies between 10 and 20, we get a 1x44 vector for each sample of the training data.
- Make the data zero-mean (subtract the mean of each one of the 44 values of the arrays from every sample

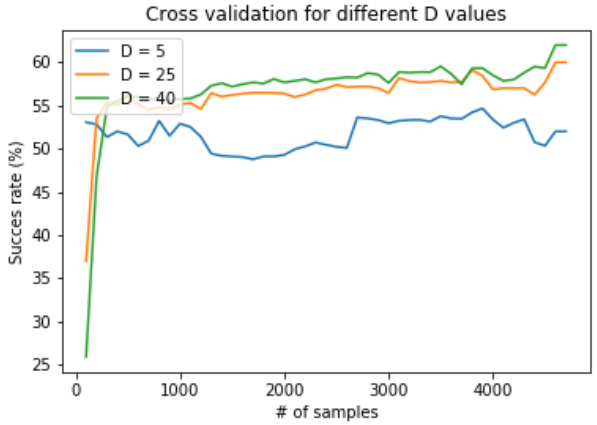

Fig. 5. Success rate for different D values for PCA algorithm.

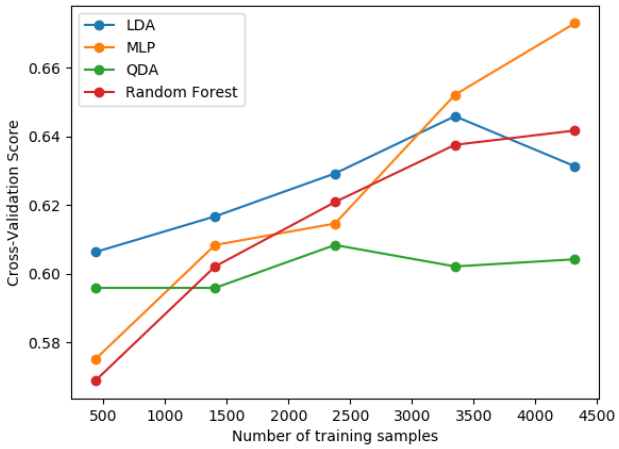

Fig. 6. Learning curve of different classifiers tested with Cross-Validation. Several classifiers were used. The Cross-Validation Score is showing the percentage of correct sample classifications.

of the training data).

- Getting a basis where all the test data will be projected on. The basis is defined as the D column eigenvectors corresponding to the D highest eigenvalues of the covariance matrix, where D is the length that the arrays will have for later processes.
- Get the covariance and the mean matrix of the separated classes, being projected on the basis first.
- Compare the likelihood of the incoming data test projected on the basis, using the covariance and the mean of each class, the data will be classified in the class that it gets the highest likelihood with.

In the figure 5 are shown some of the cross validation scores obtained for different values of D.

Each of these classifiers is performing differently for certain frequencies, as a result of this we chose the majority voting procedure to predict the final label from the predictions of all three classifiers over the last 10 trials.

Results

We observed, that the data received from each recording session is quite varying. So we have decided to use an alternative experiment layout. The best strategy for this project was to do the recording of training data, training of the classifiers and control of the robot in one session with one subject at once. The training of classifiers with data from several persons at once delivered only mediocre results. The classification got quite reliable with an ensemble of classifiers, if the mayor classified label over a time was used for the control task. The live signal is heavily varying due to distraction of the test person and eye blinking. Due to this implementation approach to minimize the influence of heavy signal deviation the time between two robot controls lasts 30 seconds. This causes a slow robot control. The current SSVEP-evaluation for robot control takes several times longer than conventional robot control with the keyboard's arrow keys. We excluded the zero label, because it disturbed classification heavily. It is important to mention, that results between two sessions are difficult to replicate. Sometimes the recorded data is good enough for the actual control task, but often the quality of data is too poor. Nevertheless, we expect a lot room for optimization regarding this issue and the duration classification.

Conclusion

We were able to successfully implement a system to use SSVEP as a Brain-Computer Interface. The SSVEP evaluation could be used to solve a control task, only by using a brain, EEG and a flickering object on a screen. During the experiments, 3 of 4 subjects felt very uncomfortable observing the flickering surfaces. The time of 10 min exposure in experiment 1 was enough, to heavily tire subjects and induce headaches. We would only recommend SSVEP evaluation, if investigations to reduce these problems are made. Nevertheless, SSVEP is a promising approach to implement BCI, because it is non-invasive and so, even casual use-cases like BCI-games could be enabled by using SSVEP-evaluation.

References

(1) Y. Wang, R. Wang, X. Gao, B. Hong, and S. Gao, "A practical vep-based brain-computer interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 234–240, 2006.

(2) C. Zhang, Y. Kimura, H. Higashi, and T. Tanaka, "A simple platform of brain-controlled mobile robot and its implementation by ssvep," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–7, IEEE, 2012.