

# Optimization Strategy for SSVEP-Based BCI in Spelling Program Application

Indar Sugiarto

Department of Electrical Engineering  
Petra Christian University  
Surabaya, Indonesia  
e-mail: [indi@petra.ac.id](mailto:indi@petra.ac.id)

Brendan Allison, Axel Gräser

Institute for Automation  
University of Bremen  
Bremen, Germany  
e-mail: [allison@iat.uni-bremen.de](mailto:allison@iat.uni-bremen.de), [ag@iat.uni-bremen.de](mailto:ag@iat.uni-bremen.de)

**Abstract**— This paper describes an optimization strategy for steady state visual evoked potential (SSVEP)-based brain-computer interface (BCI), especially in a spelling program application. In this application, there are at least three components for implementing a complete BCI application: stimulator, signal processing, and application (spelling program). Ideally, those three components should run on different processing units in order to obtain optimum performance. But integrating those three components in one computer system also gives advantages: make it easier for the subject to concentrate and simplifies the system configuration. There are two main parts that need to be optimized: the flickering animation and the spelling system. When optimizing the flickering animation, we will focus on the display driver technology and programming aspects. The optimization of spelling system will be focused on the layout and representation of the letter matrix. We tested our program on several computers for the following parameters: frequency range, frequency resolution, and frequency stability. It can be concluded that using a computer monitor as the stimulator, the maximum synthesizable stimulator frequency is always half of its minimum refresh-rate, no matter which software technology is applied (DirectX or OpenGL). The maximum synthesizable frequency of up to 30 Hz with frequency resolution 0.11Hz is achieved. We have tested our system on 106 subjects during CeBIT 2008 in Hannover, Germany. Mean accuracy for the spelling system is 92.5%. Therefore, the optimization strategy described here led to a stable and reliable system that performed effectively across most subjects without requiring extensive expert help or expensive hardware.

*Brain-Computer Interface, EEG, SSVEP, spelling program*

## I. INTRODUCTION

Brain-computer interface (BCI) systems are designed to enable humans to send commands to a computer or machine without using the normal output pathways of the brain [1], [2]. Signals from the brain, usually nonelectrical signals from the surface of the scalp (EEG activity), are classified and then translated into commands using certain digital signal processing (DSP) algorithms. To enable a signal processing system recognizing specific features of the brain signals in the cue-based BCI system, e.g. steady state visual evoked potential (SSVEP)-based BCI, a certain stimulator is required. In an SSVEP-based BCI, frequencies of the evoked potentials match the frequencies of stimuli, which are usually

elicited using flickering light [3]. In the spelling program application, the DSP algorithm translates those frequency responses into commands for controlling cursor movement and character/letter selection. The problem arises when using general purpose computers, e.g. laptops, for implementing a complete BCI application including stimulator, signal processing, and application (spelling program). Ideally, those three components should run on different processing units in order to obtain optimum performance. But integrating those three components in one computer system also gives advantages. At least, providing the stimulator, i.e. flickering light, and the application (spelling program) in the same display/screen makes it easier for the subject to concentrate and it also simplify the system configuration. These advantages are important, because end users need a BCI that does not require elaborate hardware (such as customized LED boards or separate computing systems) or expert help (such as to find working SSVEP frequencies or adapt the system to each user).

This paper will be presented in the following outline. At the beginning, software architecture will be presented and followed by its optimization strategies. Then, the software analysis result will be presented and discussed.

## II. METHOD

### A. Software Architecture

There are three components required for running an SSVEP-based BCI system in a spelling program application as depicted below:

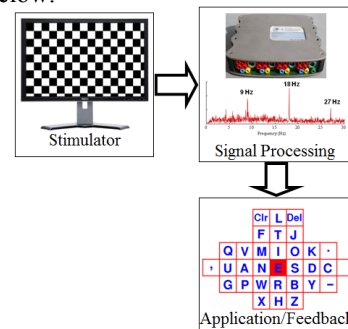


Figure 1. Three components required for running SSVEP-based BCI system in a spelling program application: a stimulator, a signal processing unit, and the spelling program. In addition, the spelling program also provides mental feedback for the BCI user.

Of course, any BCI also requires a fourth component, namely sensors to acquire the brain signal [1], but these are not modified in this work and are not mentioned further here. In this work, we used spatial filter called Minimum Energy Combination described in [4] as the core for the signal processing unit, and in this paper we focus on optimization for stimulator and the spelling program. To integrate those three components in a single computer, we created two programs working together and connected via TCP/IP socket. The first program, which is called the Display Program, is responsible for displaying flickering animation as the stimulator and also displaying the letter matrix for the spelling program as well as the visual feedback for the user. The second program, which is called the Signal Processing Program, is responsible for signal acquisition, feature extraction and classification, and command generation. Commands generated by the second program will be sent to the first program through a network connection. The following diagram shows this architecture. Both programs are written in C++.

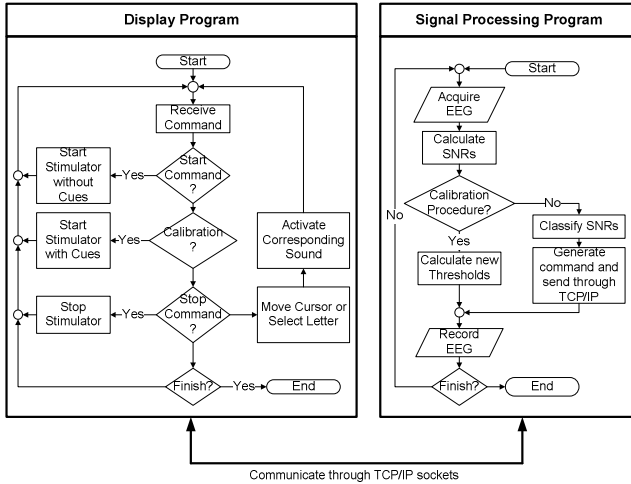


Figure 2. Three components required for running SSVEP-based BCI system in a spelling program application: a stimulator, a signal processing unit, and the spelling program. In addition, the spelling program also provides mental feedback for the BCI user.

### B. Optimization Strategies

In the Display Program, there are two main parts that need to be optimized: the flickering animation and the spelling system. The spelling system works as follows:

- We provide collection of letters arranged in a matrix and a moveable cursor within this matrix. This cursor can be moved up, down, left, or right according to the command interpreted by the Signal Processing Program from user's EEG signals.
- When the user wants to select a letter to form a word, he/she must concentrate his/her gaze on the corresponding stimulus. And if the Signal Processing Program is able to detect this intention, the letter at the cursor position will be selected and displayed on the appropriate position. In this way, the collection of selected letters will form a

word/phrase. For this purpose, we create five flickering boxes on the monitor screen with label: UP, DOWN, LEFT, RIGHT, and SELECT.

We can arrange the letters in many forms and orders. One simple way is the square matrix with ascending order as shown below:

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	CLR	DEL		-
.	,	?	!	&	;

Figure 3. One simple way to organize letters is in a square matrix. The cursor position is indicated by red-shaded cell.

The above configuration can be optimized in the following way.

- Since a word formation is achieved first by moving the cursor around, it is better if the home position of the cursor is located at the center of the matrix. In addition, since we only utilize four possible movement (up, down, left, and right), it is better if the distance of a letter from the center is kept as near as possible.
- The fact that many words composed mainly by vowels, it is reasonable to put vowels' letter such as A, I, U, E, O around the base of the cursor position. Detail analysis using letter probability of words done by Thorsten L  th [5] shows that letter E is the most commonly used letter. Based on such probability analysis, we can construct a letter matrix in an irregular but efficient way.

The following figure shows the optimization result from the aforementioned approaches. Note that this result can be expanded with additional characters if required.

			Clr	L	Del			
			F	T	J			
	Q	V	M	I	O	K	.	
,	U	A	N	E	S	D	C	
	G	P	W	R	B	Y	-	
			X	H	Z			

Figure 4. Characters are arranged in a rhombus layout in order to achieve higher efficiency for cursor movement.

The next part that needs to be optimized is the stimulator itself. From our previous work [6], it is revealed that

animations with plain texture will elicit better SSVEP response than animations with checkerboard texture. That is why we use black-and-white animation with plain texture for the stimulator.

In order to achieve robust and high resolution flickering frequencies, we developed our program with the following approaches:

- Each of five flickering boxes has its own thread and corresponding timer. Since we run the program in Windows platform, we use high resolution timer called Multimedia Timer which has resolution down to 1ms. The timer intervals are calculated as follows:  

$$Interval = 1/(2*f_{Led}) \quad (1)$$
where  $f_{Led}$  is the flickering frequency of the animation
- All of those threads are synchronized in one process and we set the process with high priority above all other Windows processes. In addition, we add CPU's mask-affinity to this process in order to utilize the first CPU's core and give the second core to the Signal Processing Program.
- Two graphics display technologies are utilized and compared: DirectX 9.0 and OpenGL 2.0. Using these technologies, our program performance is improved significantly because those technologies will maximize the utilization of the graphics card.
- When developing program with OpenGL approach, we used QGraphics Framework from QT [7]. We then optimized QGraphicsView class with the following parameters:
  - ScrollBarPolicy : ScrollBarAlwaysOff
  - Interactive : false (no interaction with other GUI events)
  - CacheMode : CacheBackground
  - OptimizationFlags : DontClipPainter and DontAdjustForAntialiasing

Here is the screenshot of our SSVEP-based BCI for spelling application.

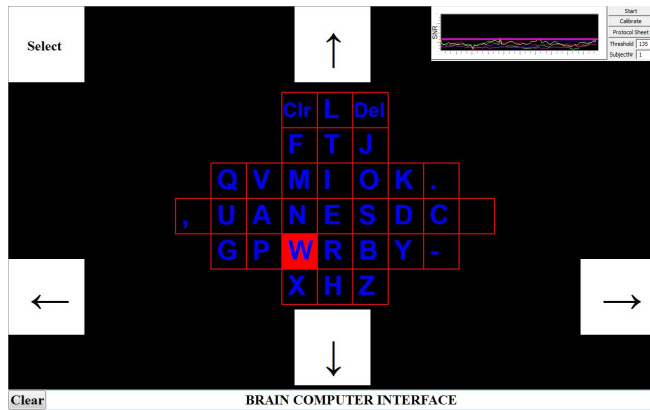


Figure 5. Integrated SSVEP-based BCI for spelling application.

### III. RESULT AND DISCUSSION

We tested our program on several computers for the following parameters: frequency range, frequency resolution, and frequency stability. We also tested our program using subjects during CeBIT 2008 in Hannover and collected useful information such as spelling speed, accuracy, ITR, and many important neuro-psychological parameters.

The following graph shows the comparison between DirectX and OpenGL approach.

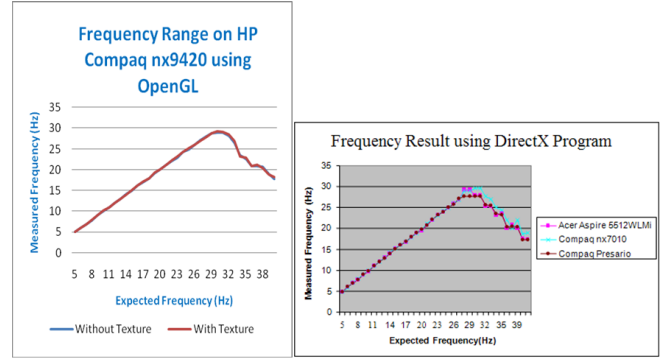


Figure 6. Comparison between OpenGL and DirectX method for displaying flickering animations on the computer screen.

It can be concluded that using a computer monitor as the stimulator, the maximum synthesizable stimulator frequency is always half of its minimum refresh-rate (for CRT monitor) or response-time (for LCD-TFT monitor), no matter which software technology is applied (DirectX or OpenGL). The maximum synthesizable frequency of up to 30 Hz is adequate since the optimum flickering frequency for low-frequency SSVEP-based BCI is around 15 Hz [8], [9].

To measure frequency resolution, first we have to measure timer accuracy during run time. In order to produce flickering animation at 17Hz, according to equation (1), we have to set the timer interval of  $1/(2*17) = 29.4ms = 29ms$  (integer value). Using Windows Multimedia Timer and executing the program as a normal priority process, we found that for timer interval of 29ms, the standard deviation value is just 0.46ms. It means one may expect that the flickering frequency produced by using this Multimedia Timer may be shifted about 0.55Hz above or below the expected frequency. This value is calculated in the following way:

$$Frequency\ shift = \frac{1000}{2\pi(29-0.46)} - \frac{1000}{2\pi(29+0.46)} = 17.52 - 16.97 = 0.55Hz \quad (2)$$

We then increased the process' priority and it yielded standard deviation interval of about 0.1ms, which means that we got the increasing frequency resolution from 0.55Hz to 0.11Hz.

To measure flickering stability against variation of animation size and number of animation objects, we conducted experiments with various CPU rating and by varying the animation size and number of animation objects

displayed simultaneously on the screen. The following graph shows the result from this experiment.

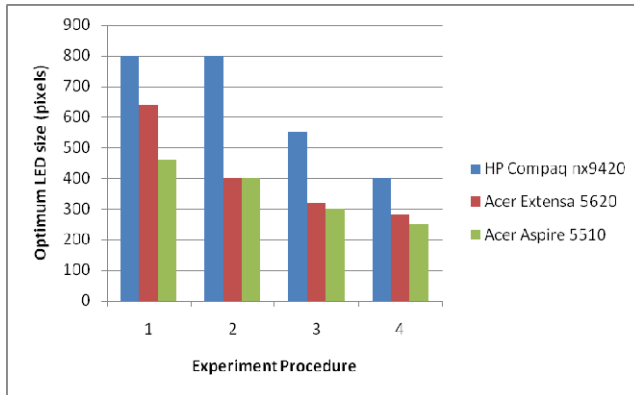


Figure 7. Comparison of optimum animation (LED) size for three computers in four different procedures: 1)Single non-textured object, 2)Single checkerboard-textured object, 3)Three non-textured objects, 4)Six checkerboard-textured objects.

It can be seen that the highest performance is achieved by HP Compaq nx9420 while the lowest performance belongs to Acer Aspire 5510. According to Windows Experience Index of Windows Vista, HP Compaq nx9420 has the value of 5.1, Acer Extensa 5620 has the value of 4.6, and Acer Aspire 5510 has the value of 3.6. It seems that the software performance, as indicated by the size of animation object as well as the number of animation object displayable simultaneously on the screen without disturbance, is greatly influenced by the computer performance.

We have tested our system on 106 subjects during CeBIT 2008 in Hannover. During the experiment, the subject was asked to spell five words: 'BCI', 'Chug', 'Siren', 'Brain Computer Interface', and one free spelling. To average subjects needed two minutes to spell 'BCI' (122.06 sec.), 'Chug' (139.3 sec.) and 'Siren' (111.13 sec.) and eight minutes to spell 'Brain Computer Interface' (475.63 sec.). Mean accuracy for all five words was 92.72%; mean ITR 13,852 bits/minute.

#### IV. CONCLUSION

The optimization strategy for the Bremen SSVEP-Based BCI in a spelling program application has been presented

and the resulting performance of the system has been evaluated. The optimization strategy in this paper is focused on the display performance of the stimulator and the speller program. We summarize the optimization strategy as follows: using advance graphics driver technology (DirectX and OpenGL), optimizing the multi-thread feature of a dual-core processor; using high resolution timer (Windows Multimedia Timer); and selection of a computer with high CPU rating. However, external factors such as light reflection and interference will also affect the overall performance of the spelling system using SSVEP-Based BCI. Future work should address these two concerns, and try to further improve SSVEP BCI performance while minimizing the needs for expert help, external LEDs, or expensive hardware.

#### REFERENCES

- [1] J.R. Wolpaw, N. Birbaumer, D.J. McFarland, G. Pfurtscheller, and T.M. Vaughan, "Brain-Computer Interface for Communication and Control," *Clinical Neurophysiology* 113, 2002.
- [2] Brendan Allison, Dennis McFarland, Gerwin Schalk, Shi Dong Zheng, M.M. Jackson, and J.R. Wolpaw, "Towards an Independent Brain-Computer Interface Using Steady State Visual Evoked Potentials," *Clinical Neurophysiology* 119, 2008: 399-408.
- [3] J. Ding, G. Sperling and R. Srinivasan, "Attentional Modulation of SSVEP Power Depends on The Network Tagged by the Flicker Frequency," *Cerebral Cortex - Oxford University Press.*, July 2006, Vol 16:1016-1029.
- [4] Ola Friman, Ivan Volosyak, and Axel Gräser, "Multiple Channel Detection of Steady-State Visual Evoked Potentials for Brain-Computer Interfaces," *IEEE Transactions on Biomedical Engineering*, 2007, Vol. 54, No. 4, 742-751.
- [5] Thorsten Lüth, "Tools zur Verbesserung der Leistung von Brain-Computer Interfaces," *Diplomarbeit, Universität Bremen*, 2006.
- [6] B. Allison, I. Sugiarto, B. Graimann, A. Gräser, "Display Optimization in SSVEP BCIs," *Computer-Human Interaction 2008*, April 2008, Florence, Italy.
- [7] Trolltech, Inc., "Creating Cross-Platform Visualization UIs with Qt and OpenGL," *Trolltech Whitepaper*.
- [8] Gary Garcia, "High frequency SSVEPs for BCI applications," *Computer-Human Interaction 2008*, April 2008, Florence, Italy.
- [9] 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*, June 2006, Vol. 14, No. 2.G.