Introduction to Computational Neuroscience

Project

Real-time frequency space visualization of BioSemi ActiveTwo EEG readings

Author: Zepp Uibo
Supervisor: Ilya Kuzovkin
Table of Contents

Introduction ........................................................................................................................................... 3
ActiveTwo device and its output .......................................................................................................... 3
Signal processing ................................................................................................................................. 3
Implementation of the application ....................................................................................................... 3
Testing ................................................................................................................................................ 4
Summary ............................................................................................................................................... 6
Sources ............................................................................................................................................... 7
Introduction

BioSemi ActiveTwo is a biopotential measurement device that is used for EEG recording. It plots and records the signal but doesn't do data analysis on it. This project's goal was to implement an application that would read the output of ActiveTwo, do real-time Fourier transform on it and visualize the results.

The main purpose for this visualization would be quick diagnostics and troubleshooting problems with the experiment setup. No real data analysis could be done with the naked eye anyway, but visualizing the results would help to recognize problems like losing input from some of the channels, or any systematic noise or bias. As setting up experiments with the EEG hat is time-consuming, having real-time feedback is also useful for running more experiments or quickly devising new ones during a lab session.

ActiveTwo device and its output

The signal from ActiveTwo is read over TCP/IP. The output from the device comes from 32 channels, corresponding to 32 electrodes attached to a test subject's head. The signal has a sampling rate of 512 Hz and 24 bit depth – that is, the signal takes values from 24 bit (8 byte) range. [1]

Signal processing

There are 2 functionalities in the program, both relying on the Fourier transform of the data:

1) power spectrum visualisation, given from the output of fast Fourier transform algorithm.
2) time – power visualization, where the user can select the band of frequencies he/she is interested in and the powers of those frequencies will be averaged and plotted over time

Both functions can be filtered by channel. For all of the channels selected, Fourier transform will be performed on the signal and the results averaged. (For the 2nd functionality the mean is taken both over all of the channels selected and over all of the frequencies in the selected band). Only the real part of the Fourier transform will be used. The output has a frequency range of 1-255 Hz; the top limit comes from the Nyquist frequency (one half of the 512 Hz sampling rate). Both results are plotted in log scale.

Implementation of the application

The application is written in Python 3, and tested on Windows 7 and Windows 8. The prerequisite libraries for running the application are:

- Python 3
- numpy 1.9.2
- PyQt 4
- pyqtgraph

The UI was developed with the help of Qt Designer, but it isn't needed for running it.

The code that reads the data from TCP/IP and writes it into a buffer shared with the main application was available beforehand, but needed to be modified. [2] The main application reads the signal (3 byte numbers) from the shared buffer, iterates over the 32 channels and does fast
Fourier transform on each of those.
The GUI consists of 2 tabs, each of which contains 32 checkboxes for selecting/deselecting the channels and a plot area below. In the 2<sup>nd</sup> tab there are also spinboxes (text inputs with up-down buttons for scrolling) for setting the frequency band.

**Testing**

For testing and presentation a testing dummy was implemented that simulates an input from the actual reader. For presentation I chose the following input: into the 1<sup>st</sup> 16 channels was written the sum of 2 sine waves with frequencies of 49 and 111 Hz. The amplitudes of the waves oscillate with a frequency of 4 seconds to simulate activity – that is, for 4 seconds the base amplitude of the 49 Hz wave is multiplied by 2, for the next 4 seconds the 111 Hz wave's amplitude is multiplied by 2, then the process repeats. Some gaussian noise was also added to all of the inputs (only noise was written into the last 16 channels). The rationale for having noise-only channels was to be able to test the channel selections.

In Figure 1 the result of the Fourier transform on this signal can be seen. There are clear spikes at 49 and 111 Hz. The power of 49 Hz signal is larger because of the amplitude multiplication described above. Because the power is plotted in log scale, the results don't show one signal being 2 times stronger, it would be if plotted in linear scale. In Figure 2 is the screenshot taken 4 seconds after the previous one. The switch in the power of the frequencies is visible.

In Figure 3, the output from the 2<sup>nd</sup> functionality can be seen. One of the frequencies of interest lies in the 49-60 Hz band selected. Also selected are half of the channels the test signal was written into and half of the channels with only noise.

![Figure 1: Output of `fft` on the test signal. T = 0s.](image-url)
Figure 2: Output of FFT on the test signal. $T = 4s$

Figure 3: Power filtered by channels and frequencies. About 1 period of oscillation in power of the 49Hz component is visible
**Summary**

At the time of writing this report the main goals were met: both functionalities were implemented and the plotting was fast. There has been one lab visit where we established connectivity to the device. If the end result is to be considered a usable and user-friendly application there remains probably about 20-25% more work (10h of programming) to be done.

Known problems and possible improvements to be made:

- During the 1st lab visit we discovered a lag between the ActiveTwo software and my implementation – the plotting runs with the same speed but in phase. It's probably a bug in the buffering code and will hopefully fixed during next visit.

- The button that selects/deselects all checkboxes works, but about 50% of the time, after deselecting all and then selecting a single checkbox, the application runs into a division by zero error in the plotting library's code. Clearing the plotting buffers and replacing zeroes with small values in the data hasn't helped.

- There's a problem with Python on Windows 8 machines not being able to fully utilize a core – on Windows 7 4-core machine the application runs at near 25% CPU (from Task Manager), but with the same configuration uses only 5-8% on a Windows 8 installation and as a result plots considerably slower. Currently the workaround is to increase the draw rate.

- The channel selectors could use channel labels instead of numbers

- The plot has a fixed size, it could be made resizable

- Draw rate could be made configurable for performance reasons on different machines
Sources