Proceedings of the 29th Annual International
Conference of the IEEE EMBS
Cité Internationale, Lyon, France
August 23-26, 2007.

SaP1D1.1

Introduction to NeuroComm: a Platform for Developing Real-Time EEG-based Brain-Computer Interface Applications

Chaunchu Wang, Haihong Zhang, Kok Soon Phua, Tran Huy Dat, Cuntai Guan
Institute for Infocomm Research, Singapore 119613
E-mail: {ccwang, hhzhang, ksphua, hdtran, ctguan}@i2r.a-star.edu.sg

Abstract—**NeuroComm is a platform to develop real time Brain Computer Interface (BCI) applications. This paper introduces the basic modules of this platform and discusses some implementation issues. With a user management module, our system is user friendly and suitable for multiple users. Also, with flexible configuration files and signal processing algorithm libraries, it is easier to integrate multiple BCI applications into one system. The NeuroComm platform also acts as a flexible tool for BCI research.**

I. Introduction

BRAIN Computer Interface (BCI) assists patients with severe physical disabilities to communicate with the outside world and to control assistive devices. In recent years, there are increasing interests in the area of BCI research and applications. Among them, BCI2000 [1] provides a general purpose research and development system. The Though Translation Device (TTD) [2] is another system. However when building real time BCI system, we realize that user friendliness is crucial for the acceptance of the system by patients. Moreover, we also found that a general-purpose platform is needed to facilitate the building of various BCI applications. In this paper, with reference to our previous works [3-9], we introduce NeuroComm, a general BCI platform for building real-time BCI applications.

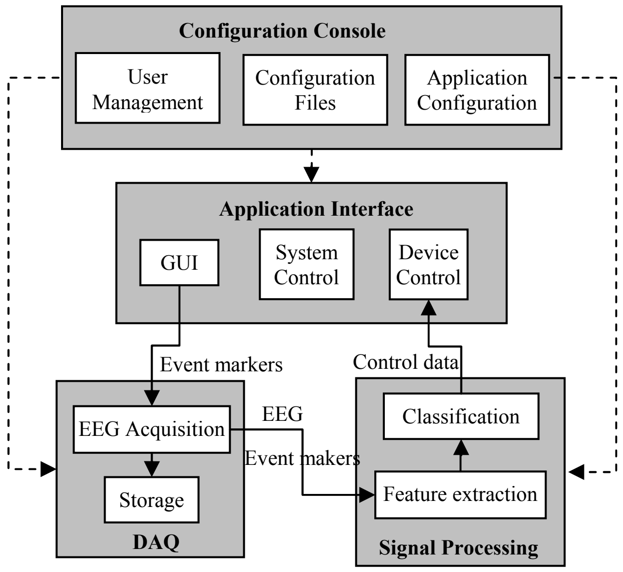


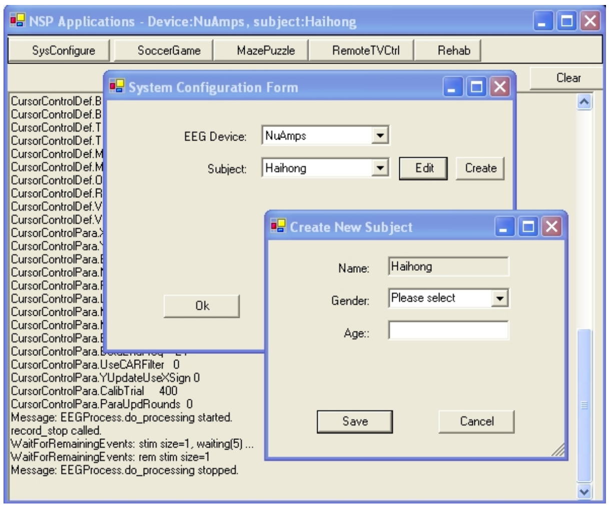
Fig. 1. NeuroComm structure. Four modules: the configuration console module enrolls new users, maintains user database, and manages system configurations for BCI applications. The application interface module has an application-specific GUI, a system control component to control the process flow and device control component to convert process result into device control signal. The DAQ module reads in EEG data. The signal processing module extracts features from EEG data and classifies them.

II. System Design

The structure of NeuroComm is shown in Fig 1. There are four main modules: application configuration, application interface, data acquisition and signal processing. The following sections will discuss all these modules.

A. Application Configuration

This module maintains a global configuration file which defines system level configurations, such as the current user, BCI applications in the system and the configuration data for each application. The console window is shown in Fig 2.



EEG Device: NuAmps
Subject: Haihong
Name: Haihong
Gender: Please select
Age::

```
CursorControlPara.UseCARFilter   0
CursorControlPara.YUpdateUseXSign 0
CursorControlPara.CalibTrial     400
CursorControlPara.ParaUpdRounds  0
Message: EEGProcess.do_processing started.
record_stop called.
WaitForRemainingEvents: stim size=1, waiting(5) ...
WaitForRemainingEvents: rem stim size=1
Message: EEGProcess.do_processing stopped.
```

Fig. 2. Application configuration console window. A toolbar displays a list of buttons. The "SysConfig" button pops up the global configuration window with functions to enroll new user and select current user and amplifier. The rest of the buttons are BCI applications that are included in the system. The message window under the menu bar displays debugging and processing messages.

1) User Management

The application console is responsible for registering new user into the system with information such as name, gender, age and other relevant information as required by the respective BCI application. When a new user is enrolled, an entry is created in the user database. Before starting a BCI application, the system operator needs to select a current user. When an application starts, the system will then loads in all the specific configuration data for the selected user. When the application ends, the user's configuration data and working environment settings are kept persistent in the system and will not be affected by other users' usage or system restarting.

2) *Configuration Files*

NeuroComm has a set of various configuration files:

- EEG amplifier configurations consisting of amplifier-specific parameters and other information needed to create the amplifier objects.
- Application configurations consisting of application specific parameters, such as task name, GUI settings etc.
- Algorithm configurations consisting of signal processing functions, such as filter design and transform matrix etc.

3) *Application Configuration*

The global configuration file has an entry listing of all BCI applications available in the system. These applications are shown as buttons in the toolbar of the application console. To configure an application into this system, follow these steps:

- Adds an application name into the "AppNames" list;
- Creates a section with the name of the application;
- Adds a "TypeName" entry with a type in the system type library as the value;
- Adds a "CfgFiles" entry with a value of a list of all app-configuration files that the application needs;
- Adds one entry for each app-configuration file, with a value of a list of consisting system configurations.

When the application starts for a new user, the system creates a working environment for this user with the necessary configurations. For an old user, if any configuration file is missing, the system automatically recreates a default one.

B. *Application Interface Module*

This module consists of three components listed below:

1) *GUI component*: It displays the application specific graphic interface, such as visual stimulus, graphic feedback etc., to the user.

2) *System control component*: It controls the application process flow, such as starting, initialization, checking processing results and stopping the system's execution, etc.

3) *Device control component:* It translates the signal processing result into control signal and controls outside devices. Here is a list of such controls:

- Cursor movement on the display screen;
- Input of a letter into a word speller window;
- Selection of TV channel;
- Switching on and off an electrical device.

C. *Data Acquisition*

This module reads EEG data from the EEG amplifier and passes them to the signal processing module. It also receives stimulus codes from the application module to construct event markers in the EEG data. It saves both the EEG data and event markers to the hard disk for off-line analysis.

There are many different types of EEG amplifiers with different specifications. Some of them are listed in Table 1. A common interface has been designed, with each type of amplifier having its own implementation and communication mechanism with the hardware. Fig 3 shows two configuration systems, each one having its own advantages. The selection of amplifier depends on the requirements of the application.

TABLE I: SOME EEG AMPLIFIERS USED BY NEUROCOMM

Amplifier	Specifications
Nuamps	40 channels, portable, power on USB, USB and Scan4.3 for DAQ, QuickCap, trigger input, www.neuroscan.com
SynAmps2	66 channels, separate power unit, USB and Scan4.3 for DAQ, Quickcap, trigger input, www.neuroscan.com
BioRadio150	8 channels, wireless, USB and c++ API for DAQ, no trigger input, www.clevemed.com
g.USBamp	16 channels, power on AC, USB and c++ API for DAQ. No trigger input, www.gtec.at

All information here can be found from the manufacturer's webpage.

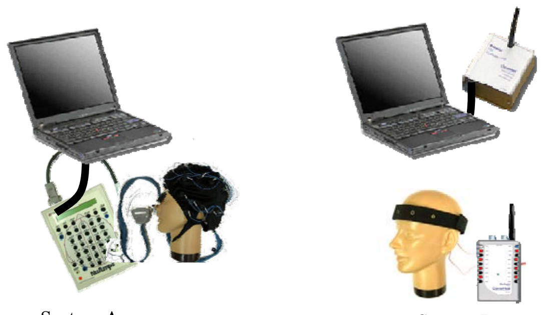


Fig. 3. Two EEG configuration systems. System A uses Nuamps, an EEG cap and a computer; while system B uses BioRadio150, and head band with electrodes attached and a computer. The advantage of system B is that the user is not bound to the computer, he/she can move freely because the user unit is separated from the computer unit, and they communicate with each other through wireless radio. On the other hand, system A has more EEG channels with higher resolution.

Two virtual amplifiers are built in this module: one is the Simulator which simulates a real amplifier by reading data from hard disk file; another one is the Live-Simulator, which is similar to the Simulator except that it follows the speed of the real amplifier when reading EEG data. With virtual amplifiers, we can repeat the application execution offline, which is useful for offline debug and study.

D. *Signal Processing*

This module processes the EEG data, extracts signal features, classifies these features using pattern recognition and machine learning algorithms ([4-9]) and passes the processing results to the application module to be translated into device control signal.

1) Preprocessing

A set of preprocessing algorithms has been used to the raw EEG data, different for particular BCI applications. They are: filtering both in frequency domain and in space domain, down sampling, PCA, etc. Filter-Bank technology has also been used in some applications [7].

2) Feature extraction

Different features for different application have been extracted for different classifier. For example, in the P300 based speller system [8], the down-sampled EEG data together with delta transform results have been used as feature in a SVM classifier. In [7] the entropies of selected frequency bands are used as the feature in the motor imagery detection.

3) Classifiers

Classifier can be simple or complicated depending on the specific BCI applications. In a mu-beta rhythm based cursor control system, the mu and/or beta frequency bank energy power from different EEG channels are used to calculate the steps of the cursor movement, with a calibration program training all the parameters. For a P300 based speller system, we have used SVM as the classifier. For asynchronous applications, the idle state where the user is not doing specified tasks should also been identified. In [4] a statistics model is proposed for an asynchronous speller. We are still working on asynchronous motor imagery tasks.

III. Some Implementation Issues

The system is implemented in C++ and C#. The DAQ module and the signal processing module are written in C++, because most EEG amplifiers are bundled with Windows drivers and C++ interfaces, and most signal processing algorithms are written in C++. On the other hand, the interface module is written in C# because of its easiness and richness in GUI design. In this section, we will discuss some of the problems we faced and the methods we used to solve them. In [3], we have already discussed the implementation issues on the synchronization between different modules and the accurate timing problems. As such, this paper will concentrate on other problems.

A. Event markers

Event markers are used to mark special events such as user responses and system cues (visual or auditory) during data collection. They are integer values forming a special channel in the EEG data. At the signal processing stage the EEG data can be segmented and labeled using these event markers.

We note that some EEG amplifier has hardware triggering mechanism to generate event markers. For example, Nuamps and SynAmp2 have trigger input ports which interface with a PC with a 25-pin parallel port. However, in Nuamps, we found that when the stimulus is sent at a frequency of 40Hz or higher, some event markers are lost. For other amplifiers, such as BioRadio150 and g.USBamp, do not have hardware trigger inputs. As such, we need to either modify one EEG channel for trigger input, or use software to generate these event markers. The following are some of the solutions.

1) Detection and recovery of lost hardware event markers

For amplifiers with hardware trigger input ports, a mechanism has been implemented to detect and recover lost event markers. By keeping a queue of stimulus codes received from amplifier module, the system will check the event markers read from the EEG amplifier. Whenever a mismatch occurs, a lost event mark is identified. An error processing program is designed to recover this type of error by inserting the lost stimulus code or report an error and interrupt the system execution based on to the system configuration.

2) Software generation of event markers

For EEG amplifiers without trigger input, an algorithm has been designed to generate these event markers. One implementation is that when a stimulus code is received, the system inserts it into the event marker's queue. This method may encounter problem if accurate timing of triggering is required, because the EEG data is read in blocks; each block usually consists of multiple samples. For example, for an amplifier with 250Hz sampling rates and block size of 10 samples, one block will be 40 milliseconds. If the stimulus codes are sent every 30 milliseconds, the generated event marker queue will not be accurate (see Fig 4).

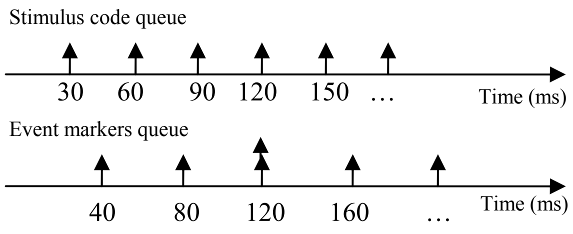


Fig. 4. Simple event markers generation algorithm could have problem. Stimulus codes are sent every 30 ms, amplifier sampling rate is 250Hz, and EEG data reading block size is 10 samples. The resulting event marker queue will not be accurate, and at point of 160ms, there will be two event markers.

A more accurate algorithm is to use the time duration between consecutive stimulus codes to determine the number of samples between the relevant event markers.

B. Communication between modules

1) C# calls C++ functions

We have created a dynamic link library to export C++ functions of modules written in C++ which was later imported into C#. Two steps are to be followed as the following:

- Create proxy classes in C# to represent modules written in C++, defining all the interfaces that C# to call.
- Create unmanaged C++ dynamic link library to

implement and expert all the C++ functions that are to be called by C#.

2) C++ calls C# functions

An operation scenario whereby such C++ calls is needed is when the application operator needs to check the intermediate results in order to adjust the parameters in the system configuration. This adjustment will require modules written in C++ to pass data to C# for display. It is not practical for C# to call C++ functions to get the data because the application interface does not know when the data will be ready; chances are that it may miss some data. Fortunately C# has a delegate type which can be passed to C++ as a function pointer.

- C# defines a new delegate type according to the function to be called by the C++, creates an object and passes it to C++ as a function pointer;
- C++ calls the function referenced by the pointer to pass data to application interface in C#;
- C# displays intermediate results.

C. Debugging message and system logs

In fig 2, there is a message window that displays system message such as debugging information and processing statuses. Such messages are saved onto a system log file for offline inspection. Our mechanism makes this display window transparent to all modules. In other words, these modules only need to write the debug and process messages to the standard output stream which will subsequently be displayed on the message window. We have used the C++ pipeline and file redirection methods to capture standard outputs both from C++ and C# modules and redirected to the message window and the log file. This mechanism helps to achieve the independences of different modules.

IV. Conclusion

NeuroComm is platform that is flexible for many BCI applications to be included into one single system using configuration files. With this platform, we have created a number of applications. Fig 4 shows four BCI applications with the configuration console which shown as Fig 2.

a) A soccer game based on the mu-beta rhythm from the subject's raw EEG data. System moved the soccer upwards automatically while the user coordinates the mu-beta rhythms to control the movement of the soccer in the horizontal direction so that it can hit the goal.

b) Maze game. User controls the movement of a bird to eat the seeds (pink dots) along the paths. Three eagles (blue, yellow and brown) walk in the maze and choose path randomly. The bird should avoid the eagles so that it will not be eaten by them. User will be awarded if he/she has successfully controlled the bird to eat all of the seeds with limited number of lives.

c) A virtual TV remote controller. The buttons on the controller flashes one by one randomly. User selects a button by gazing on it for a certain period.

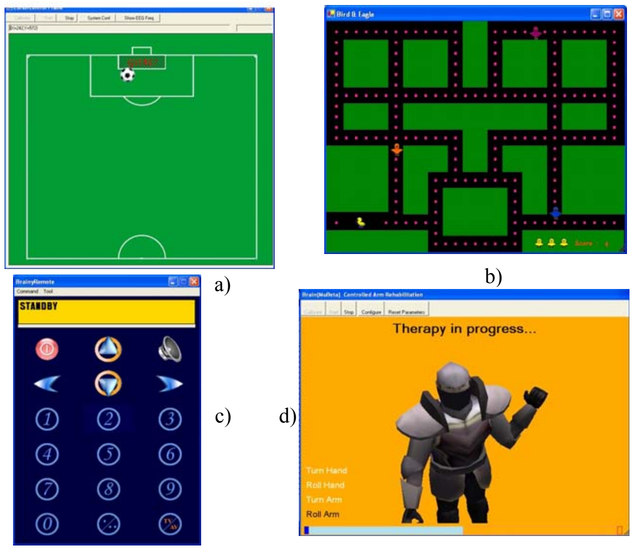


Fig. 5. BCI applications in one system. a): Soccer game; b): Maze game. c): Virtual TV controller. d): Brain control of rehabilitation process.

d) Brain control of rehabilitation process. System highlights the rehabilitation processes one by one; user actively chooses a rehabilitation process through motor imagination.

References

[1] Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, Jonathan R.Wolpaw, "BCI2000: A General-Purpose Brain-Computer Interface (BCI) System," *IEEE Transl. on biomedical engineering,* vol. 51, No. 6, June 2004, pp. 1034-1043.

[2] N. Birbaumer, A. Kübler, N. Ghanayim, T. Hinterberger, J. Perelmouter, J. Kaiser, I. Iversen, B. Kotchoubey, N. Neumann, H. Flor, "The Thought Translation Device (TTD) for Completely Paralyzed Patients", *IEEE Transactions on rehabilitations engineering,* vol. 8, No. 2, June 2000.

[3] C. Wang, C Guan, H. Zhang, "P300 Brain-Computer Interface Design for Communication and Control Applications," *Proceedings of the 2005 IEEE Engineering in Medicine and Biological Engineering 27th Annual Conference,* Shanghai, China, September 1-4, 2005.

[4] H. Zhang, C. Guan, C Wang, "A statistical model of brain signals with application to brain-computer interface," *Proceedings of the 2005 IEEE Engineering in Medicine and Biological Engineering 27th Annual Conference,* Shanghai, China, September 1-4, 2005.

[5] X. Zhu, C. Guan, J. Wu, Y. Cheng, and Y. Wang, "Expectation-Maximization Method for EEG-Based Continuous Cursor Control," *EURASIP Journal on Applied Signal Processing*, Volume 2007, Article ID 49037.

[6] Thulasidas Manoj, C. Guan and J. Wu, "Robust Classification of EEG Signal for Brain-Computer Interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 14, No. 1, 2006, pp 24-29.

[7] Tran Huy Dat, H. Zhang, C. Wang and C. Guan, "Selected Subband Entropy for Motor Imagery Detection in Asynchronous Brain Computer Interface," to be presented on *The 3rd International IEEE EMBS Conference on Neural Engineering*, 2-5 May, 2007, Hawaii, USA.

[8] H. Zhang, C. Guan, Y. Li, "Signal Processing for Brain-computer Interface: Enhance Feature Extraction and Classification," IEEE International Symposium on Circuits and Systems (ISCAS), Kos, Greece, May 21-24, 2006.

[9] Manoj Thulasidas, Cuntai Guan, "Accurate and Fast-Learning Classification of P300 Potential for a Real-Time Speller," *27th International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS),* Sept 1-4, 2005, Shanghai, China.