

Temporal Analysis of EEG patterns in a bio-feedback based Brain Computer Interface

David Starling

BSc Computer Science and Cybernetics, davestartling@me.com

Abstract – This paper describes the development of a brain computer interface. The system allows the user to control a game character in a 3D virtual environment, as well as control a basic 'Pong like' bar in a simple 2D application. The user will have to adapt their thought patterns to be consistent, and to be able to successfully control the game. As the user moves through the game, various methods of bio-feedback are used to allow the user to understand their performance, including rewards for successful movement, and visual and aural 'punishments' for unsuccessful movement. The user input comes from electrodes placed on the skull, which are connected to an EEG (electroencephalogram) machine, which in turn is connected to the PC via an ADC (analogue to digital converter) card.

Contents

1. INTRODUCTION.....	3
1.1. BACKGROUND.....	3
1.2. BCI SETUP AND EEG STATES	4
1.3. USER TRAINING.....	6
2. HARDWARE	6
2.1. THE GRASS-TELEFACTOR EEG MACHINE	7
2.2. THE PC.....	8
3. SOFTWARE.....	9
3.1. FEATURE EXTRACTION.....	10
3.2. FEATURE CLASSIFICATION	12
3.4. TRAINING ENVIRONMENTS	13
4. EEG DATA SAMPLING	15
4.1 MENTAL TASKS AND DATA RECORDING	15
5. RESULTS	15
5.1. TESTING THE EEG MACHINE.....	15
5.2. TESTING THE RARX FEATURE EXTRACTION	16
5.3. TESTING THE PNN FEATURE CLASSIFICATION	16
5.4. TESTING THE TRAINING ENVIRONMENTS	16
5.5. TESTING THE EXTRACTION AND CLASSIFICATION ON EEG DATA	17
5.6. TESTING THE SYSTEM	18
6. FUTURE WORK	18
7. CONCLUSION.....	18
8. ACKNOWLEDGEMENTS.....	19
9. REFERENCES.....	20

1. Introduction

Brain computer interfaces are a relatively new field of study, but is rapidly advancing to the point where users will be able to operate a computer or other device by thought, rather than physical or verbal input, or nerve signal interception. This paper describes the development of, and the techniques involved in, a brain computer interface (BCI). Such a system has many applications, since the output can be used to control any arbitrary device that requires an input – some of the more recent applications include those that give physically disabled persons the ability to use robotic limbs, and further immersion in virtual reality systems.

The users of the system will be able to produce simple control signals via their thought patterns (for example, imaginary hand movement, mathematical calculations, and geometric manipulation). These control signals will allow the user to interact with specific software on the computer.

The system described in this paper is designed to allow the user to control one of two applications: a basic 2D application that indicates the output the user is producing via a graphical bar, and an immersive, 3D game that will give much more information rich bio-feedback – both visual and aural. This bio-feedback will allow the user to see how well they are producing specific, consistent, thought patterns, and adjust them accordingly.

1.1. Background

Almost all current computer input devices require some physical contact between the user and the input device, for example a keyboard, mouse, or trackball. Other interfaces might include speech recognition to allow computer tasks and dictation to be performed by the user without any physical contact. Despite the wide range of computer input devices, it is clear that there may be circumstances where current devices are not suitable. For example, if the user has a motor-control disorder both speech and physical input may be impossible. An alternative input which this paper investigates and develops involves reading the mental states of the user by studying the electrical neuron activity of their brain.

The concept of interfacing a computer directly to the brain is a relatively new one, but analysis of brain waves has been occurring since 1929 when Hans Berger first recorded EEG signals from a human source using electrodes and a galvanometer. This measured the potential difference between the electrodes placed on the scalp and a reference electrode, placed on the earlobe. The potential difference fluctuations gave an indication to the small current produced by the billions of neurons in the brain when they fire. The combination of the neurons firing in the brain can yield extremely complex signals [8], but with identifiable patterns based upon the activity in the brain. For example, sleeping will produce a substantially different EEG signal to that of a brain computing complex mathematical problems – this can be seen by examining the frequencies of the signals. It has been demonstrated in previous BCI attempts that the isolation of particular features of interest can be extracted [9, 10]. Through training, the user can learn to control these detectable features. This yields the possibility of using these voluntary thought patterns as control signals for use in a system.

Most current brain computer interface designs use an electroencephalogram to measure the potential difference between electrodes placed on the skull. The EEG machine is standard medical imaging device, and has many advantages - it is non-invasive, safe, and relatively easy to use. Other techniques include intra-cortical electrodes [11], which although produce signals with less noise, require surgery. However, successful attempts at this have used

patients that already have the electrodes from epilepsy monitoring trials. One of the key roles of the EEG machine is signal amplification. The signals received from the electrodes are minute, and to generate a useable signal, they must be amplified substantially. Other systems might involve the insertion of electrodes or other devices directly into the brain, and indeed, some research has shown that this is a more effective method due to the lack of interference from the skull itself. Effective electrode placement was one of the key areas researched, and has many practical considerations. Although the signals from the EEG themselves are complex, various analysis and filtering techniques can be developed to reduce the complexity of the signal, and extract and classify key features.

One of the key features of a brain computer interface is the seemingly limitless number of applications. One of the main application areas is that of prosthetic limb control, allowing a paralysed patient to gain motor ability in a BCI controlled limb. Potential users include those with severe head trauma or spinal injuries, as well as people with amyotrophic lateral sclerosis (ALS, or Lou Gehrig's disease) or cerebral palsy.

Most prosthetic limbs make use of voluntary motor control – i.e. they intercept the nerves that would otherwise be sent to the limb - and as such are not always practical. The EEG machine measures brain patterns, and is not reliant upon nerve signals that may have been severed. It is therefore possible to perform real time analysis of EEG components during imaginary hand movement by measuring the potentials above the motor cortex. For completely paralysed users – for example, those with amyotrophic lateral sclerosis - this makes prosthetics a possibility, and gives the ability to interact with the environment.

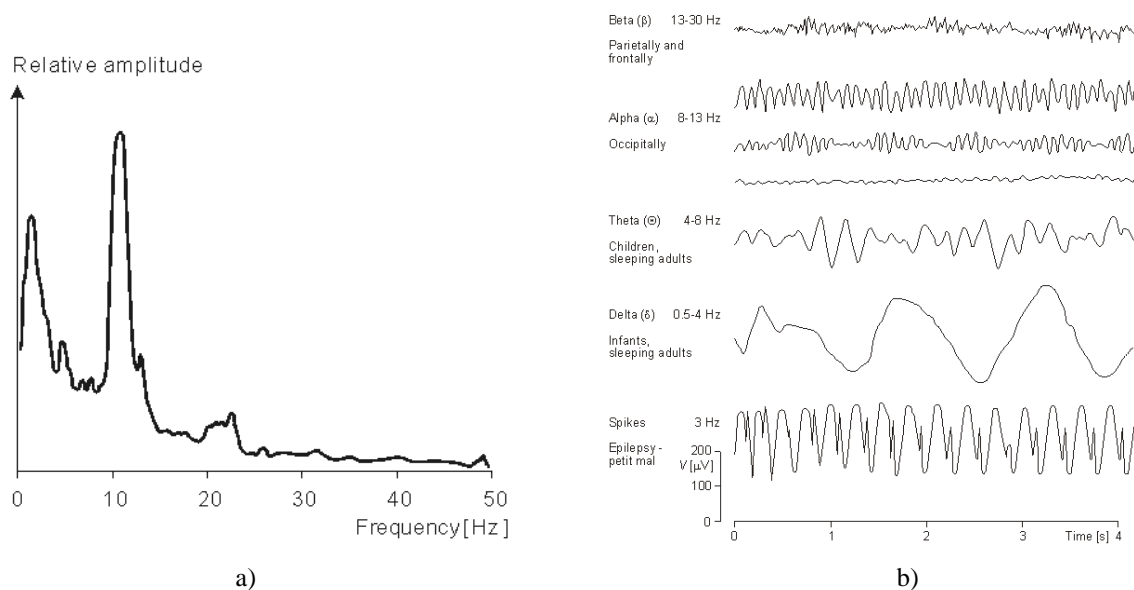
When using scalp electrodes to measure the EEG patterns a number of filters and pre-processing need to be performed to remove noise created by mains (50Hz) and monitor frequencies (60-85Hz). The impedance between the scalp and the electrodes is also a key factor and early EEG recordings where the results were recorded on paper and required an impedance of 5k Ω [12]. However, with digital recording of the data it has been found that an impedance of 40k Ω will still produce usable results.

1.2. BCI Setup and EEG States

To be able to successfully recognise the mental tasks being performed by users, various EEG states need to be analysed. By examining these characteristics it is possible to get a general view of the mental activity and state of mind of the user.

Table 1. EEG states of the mind

EEG State	Associated Frequencies	Associated Mental States
<i>Alpha</i>	7.5-13Hz	Alpha waves are associated with relaxation and having eyes closed. The amplitude of alpha waves ranges between 10 and 50 mV.
<i>Beta</i>	13-40Hz	Beta waves are associated with alertness, arousal, problem solving, and concentration. Beta waves are fast but low amplitude.
<i>Delta</i>	0-4Hz	Delta waves are associated with deep sleep. They are a high-amplitude, low-frequency wave, and are generated by the lack of processing by neurons. Delta waves can also be found when examining a comatose patient.
<i>Theta</i>	4-7Hz	Theta waves are associated with sleep, but can also be associated with anxiety, epilepsy, and traumatic brain injury.
<i>Mu</i>	7-11Hz	Mu rhythms are associated with the motor cortex, and can be used to recognise imaginary motor movement.

**Figure 1.** An example EEG waves showing a) the typical frequencies in an EEG wave, and b) the specific EEG states

EEG waves can also contain short-term transient waves known as event-related potentials, which reflect a singular experience associated with an external stimulus. This causes a small voltage fluctuation within the EEG wave.

Although frequency analysis allows the recognition of the general state of mind, it does not allow recognition of specific mental tasks. It therefore cannot be used for precise control of a BCI. However, it was realised later on in the project that frequency analysis could be used to allow better recognition of tasks when compared to a baseline, resting, state.

In the early stages of BCI research, it was unclear whether motor imagery could be easily identified from background EEG signals, and how adaptive from person to person it would

be. Penny et al.[1], however, reported the recording of EEG signals from seven people who performed imaginary hand movement. They then used Laplacian operators to estimate local activity on three sites each side of the sensorimotor cortex. Analysis of the mu-rhythm in the signals showed that imagined movement could be detected in six out of the seven people, with around 70% accuracy [2].

Recent BCI implementations, such as that by Christoph Guger et al [3], have involved a seven-stage system (Fig 2), involving the user, the EEG, signal capture via the ADC, signal filtering, feature extraction, feature classification, and a resulting action. The actions in our system will be in the form of visual and/or aural feedback.

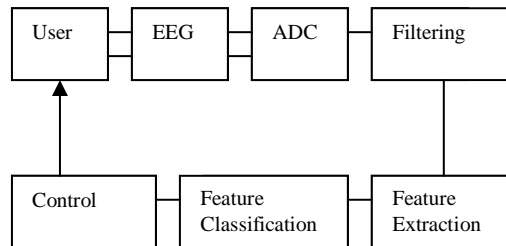


Figure 2. The general BCI system used for this project

1.3. User Training

In current BCI systems, the setup requires training of the user to control their mental state to be consistent and allow control. To date, two main approaches have been used. The first method involves training the subject by instructing them to perform specific cognitive tasks. These mental tasks can range from imaginary motor movement (one of the most commonly used cognitive tasks in BCI studies) and non-trivial mental arithmetic, to tasks involving mental 3D rotation of an object and other spatial tasks. Tasks involving imaginary motor movement can be recorded by electrodes placed above the motor cortex in the central head region, and other studies have produced results that show control can be achieved this way.

The second approach involves bio-feedback, or more recently neurofeedback, and gives the user much more freedom to perform the task. The user may produce any kind of brain pattern, as long as they can gain control of the system. The bio-feedback provides a system for the brain to adapt to control the EEG signals in a way that will allow control of the system in a defined manner. Over a series of training sessions the user gains complete control of the system, but isn't consciously thinking of specific commands. This has been compared to the skill of riding a bike [13], which is not a conscious ability.

The types of neurofeedback given to a user is any area that hasn't been examined in much detail, and the BCI system developed in this paper is designed to allow a direct comparison between basic feedback systems and information-rich feedback systems.

2. Hardware

The system's hardware involves only two main devices – the Gras EEG machine, to which the electrodes are connected using the international 10-20 electrode system (Fig 3). This in turn is connected to an ADC card within the PC.

2.1. The Grass-Telefactor EEG Machine

The Grass-Telefactor electroencephalogram machine (Fig 4) is a standard medical device, and uses an array of electrodes attached to the user's scalp. The user's scalp is first prepared with an abrasive paste to remove any dead skin and sweat which may interfere with the signal. The optimal impedance between the scalp and the electrode is $5k\Omega$ [12]. Each gold electrode has a small amount of conductive paste applied to it, which is then placed underneath a cap in the area of the skull we desire. Early development involved us attaching these electrodes to the forehead, to allow electrooculogram (EOG) eye muscle signals to be detected and used to test the basics of the system. The output signals from the EEG machine contain the filtered and amplified potential differences between each electrode and the reference electrode. Common reference electrode positions are Cz, earlobes, mastoids, and the tip of nose. It was decided that the reference electrodes for this project should be placed on the mastoids. Due to hardware constraints, only four other electrodes were used. Once the mental states the project was examining were decided, the electrode positions were chosen to be C3, C4, P3, and P4. This would allow the system to recognise both imaginary motor movement tasks, as well as tasks such as mental arithmetic, counting, and 3D rotation. A cap was purchased and labelled with the correct electrode placements. This allowed much tighter grip of the electrodes than earlier testing where the electrodes were taped to the user with micro-pore tape. It also allowed more consistent positioning.

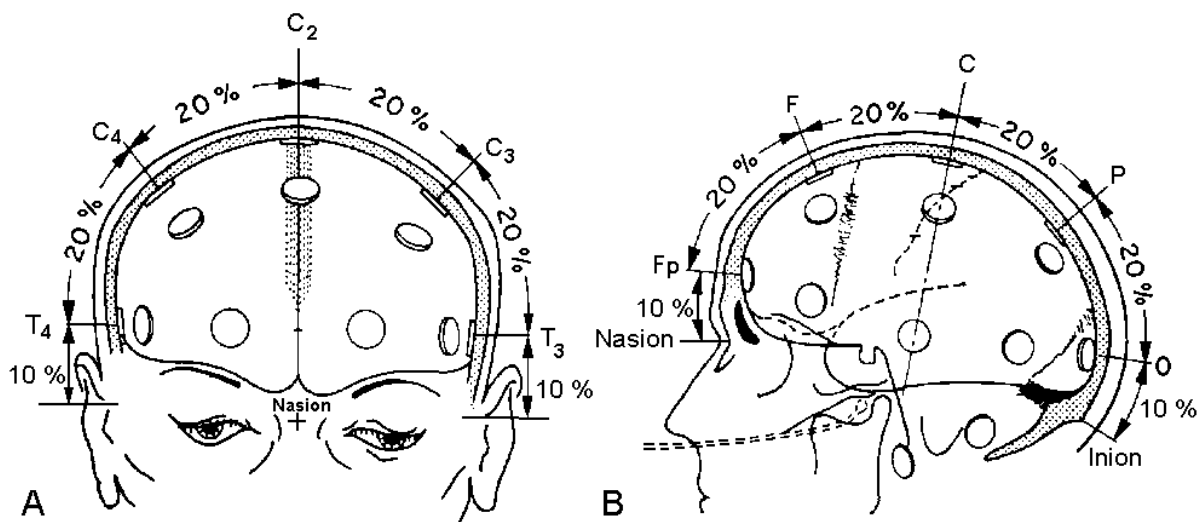


Figure 3. The international 10-20 electrode system, showing the C3, C4, P3, and P4 electrode placements.

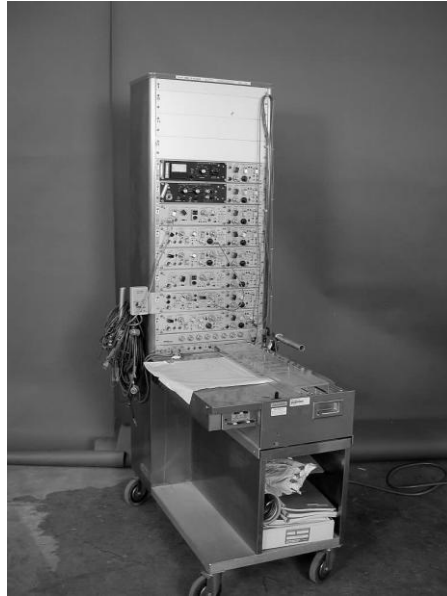


Figure 4. The Grass-Telefactor EEG Machine

2.2. *The PC*

The PC provides the system with the required data capture and analysis hardware, as well as a platform with which to give the visual and aural bio-feedback. A dedicated PC was procured, which consisted of a 1.1Ghz Duron processor, with 128MB RAM. However, once significant progress had been made with the software development, it became apparent that this hardware would not provide adequate performance for our needs. With both complex signal analysis and graphical feedback being run on the same computer, the performance became too poor. We therefore upgraded the machine to a 1.6Ghz Pentium 4, with 256MB RAM. This provided much more adequate performance, and allowed the running of both signal analysis and feedback software on the same machine.

Given more time, and a larger budget, two computers would have been used. When networked together, each computer would have a dedicated task of either signal analysis or graphical feedback.

The PC contained a high-performance ADC card, a PCI DAS-1000, to allow real-time capture of the EEG signals. The EEG machine itself has outputs that can be fed into another device, and a screw-board allowed us to interface the two together. This was then attached to the ADC card via a 100 wire ribbon cable.

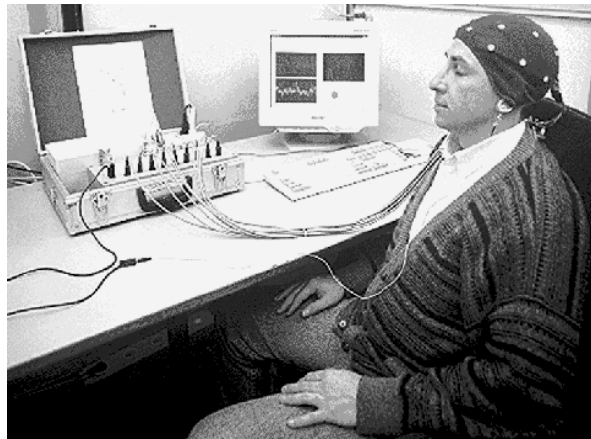


Figure 5. A typical EEG setup.

3. Software

The EEG signals themselves can be classified according to the area of the brain. According to Guger et al. [1], the imaginary movement of one hand will lead to an alpha band rhythm desynchronisation in the opposite side of the brain, and in some cases synchronisation of hand-eye coordination related rhythms in the same side of the brain. It can be deduced that an array of electrodes placed over the somatosensory cortex (Fig 6) will lead to useable signals from the EEG machine. However, to actually use these signals effectively, reliable analysis of the EEG signals is required.

Research into signal analysis methods showed that a three stage method is necessary. The first stage involves filtering, and noise reduction. Specific frequencies, such as 50/60Hz mains, are removed and then fed into the second stage. The second stage involves feature extraction. This stage involves reducing the complexity of the EEG signals. This allows much faster classification because the signal is effectively compressed, and only the relevant and useful parts are used.

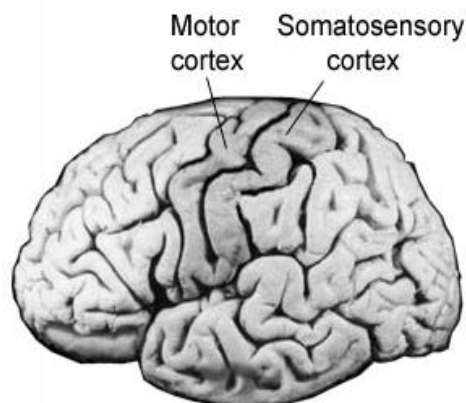


Figure 6. The somatosensory cortex - the area of the brain responsible for sensory perception.

Mathwork's MatLab software was used for implementation of all the signal analysis, and interfaced with two external programs – the 3D environment written in Java, and the 2D application, written in C#. Several methods of interfacing the software were examined, and developed. The first method involved MatLab generating keystrokes, using a DLL written in

C++, and initial results proved promising. However, when implemented in Simulink – MatLab’s real-time modelling software – the software caused the machine to hard reboot. It was then decided that the software could be interfaced using Dynamic Data Exchange (DDE). MatLab has inbuilt DDE functions, so the 3D Java application was modified to run a DDE server instead of listening to keystrokes.

After several more failed attempts at implementing a system in Simulink, it was decided to move the platform to use plain M code within MatLab, and use the Data Acquisition Toolbox to gather the data for both offline and online analysis and classification. This gave several advantages, as the ARX and PNN had already been written in M code and integration would be much faster. The DAT also allows ‘callback’ routines to allow functions to be executed at either specific times, or after a specific number of samples have been recorded. Early tests ran the TimerFcn callback routine, which ran the extraction and classification routines every 0.1s. However, to work correctly, ARX needs to be executed as every new sample is received. The SamplesAcquiredFcn callback routine was used, with the SamplesAcquiredCount parameter set to 1. This allowed the ARX function to be executed after every sample. In order to keep the system responsive, the classification was not run after every sample.

3.1. Feature Extraction

In order to allow for less computation time in the classification stage, a feature extraction stage is added prior to it. The reducing the complexity of the signal, by compressing it and extracting only the relevant and useful parts of the data, the computation time can be drastically reduced [4]. Parametric modelling was chosen for this task. Parametric modelling is a technique by which a mathematical model is fitted to a time series, and the time varying properties of the signal can be analysed. Several PM methods were examined for their effectiveness. These included auto-regression (AR), adaptive auto-regression (AAR) – the online version of AR, auto-regression with exogenous inputs (ARX) and recursive auto-regression with exogenous inputs (RARX). All these methods allow for a small number of electrodes to be used, whilst still yielding useful results.

Guger’s paper [1] showed the design and production of an AAR based BCI controlled prosthesis, where the AR coefficients are estimated using Recursive Least Squares. The inputs to this AAR model were from two bipolar channels placed on the C3 and C4 areas international electrode system (Fig 3), sampled at 128Hz. The AAR model allows an EEG signal to be described using very few coefficients, and by following the values of these coefficients, the signal’s time varying properties can be traced. These coefficients can then be used to classify the EEG signal.

This system uses an RARX feature extraction model, which allows online (real-time) analysis of the system parameters, and each iteration that updates the output coefficients can be calculated with minimal computing overhead. The system treats the non-stationary signal as a model plus Gaussian noise, and is given a list of start model parameters and the EEG signal for the previous n samples, where n is the order of the model. These parameters are the adjusted by the algorithm, based upon the error between the parameters and the EEG signal. The RARX implementation used also allows for data from much earlier on in the data to be ‘forgotten’, using a forgetting factor. This assumes that data from some time ago is considered less important than the most recent data. The forgetting factor was found to be required because the dynamic model parameters would occasionally get ‘stuck’, even though the signal is changing substantially.

A system using ARX can be represented as:

$$y_n = \Phi_n \theta_n \quad (1)$$

where y_n is the output at time n , Φ_n is a collection of previous data in the time series and θ_n is a vector of coefficients - i.e. the parameters of the model. These coefficients can be estimated using:

$$\hat{\theta}_n = (\Phi_n^T \Phi_n)^{-1} \Phi_n^T y_n \quad (2)$$

As more data is acquired, the coefficients are calculated recursively. To perform this task efficiently, we have $\hat{\theta}_n = \theta_n + \text{some correction factor}$ where the correction factor is based on the prediction error of the current estimation.

To calculate these coefficients, a Recursive Least Squares (RLS) method is used. This recursive model estimation (which might be better considered as iterative) allows estimation, albeit poor, from the first time step. It is also computationally efficient, and does not have large memory requirements. This is a major advantage in an online application that uses graphically intensive training.

The RLS method calculates the error of the estimated model when compared to the simulated system and adjusts the coefficients. This prediction error defined as the difference between the actual output and the output given by the previous model and the new data, and is calculated as follows:

$$\varepsilon = y_n - \varphi_n^T \theta_{n-1} \quad (3)$$

The coefficients can then be updated:

$$\hat{\theta}_n = \hat{\theta}_{n-1} + K_n \varepsilon_n \quad (4)$$

where K_n is the scaling (otherwise known as the Kalman Gain).

$$K_n = P_n \varphi_n \quad (5)$$

$$P_n = (P_{n-1}^{-1} + \varphi_n \varphi_n^T)^{-1} \quad (6)$$

Since this system is being used for online classification of mental tasks, it is clear that data that occurred some time ago is less important than more recent data. A forgetting factor λ , can be implemented to allow this, and indeed it was found that without λ the dynamic model parameters would get stuck upon significant signal changes. In order to modify the parameter estimation so that older data has less impact, the squared error is biased.

$$\sum_{i=1}^n \lambda^{n-i} \varepsilon_i^2 \quad (7)$$

This gives the sum of the squared errors where n is the number of data points and i varies with the age of the data.

Once the ARX function had been written, it was tested using a stable AR signal produced by MatLab's *filter* function, which allows the generation of a stable signal when passed model parameters.

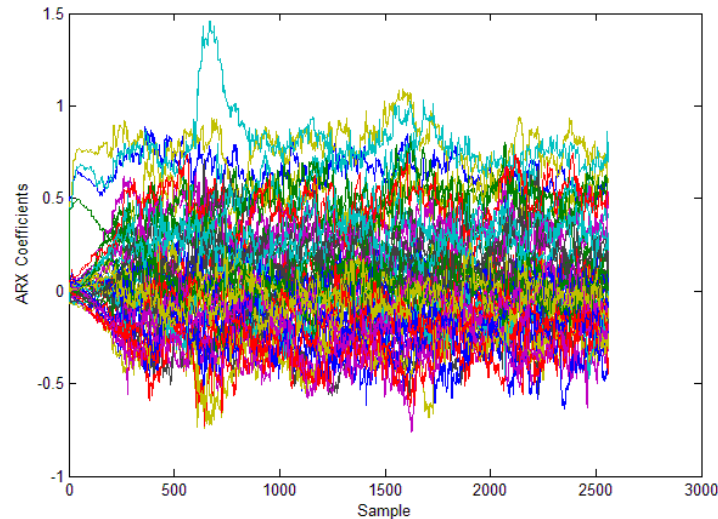


Figure 7. An example of 48 ARX coefficients changing over a 20 second sample of EEG data.

The order of an ARX model is defined as the number of coefficients used to estimate the system. Other groups had discovered, mostly through trial and error, that an order of twelve was ideal for EEG modelling [15]. This led to an overall system using forty-eight coefficients - twelve for each of the four channels. An example of the coefficients updating over time can be seen in Figure 7.

3.2. Feature Classification

When it came to deciding upon a classification method for the signal features there were two main classifiers examined – a neural classifier, and linear discriminant analysis (LDA). An LDA weights each input parameter according to its importance [5]. The sum of the weighted parameters gives the classification result, and an indication to which class the input belongs.

A neural classifier learns by examining previous known data, and generalising new data that may not have been seen before [6]. This gives some room for the system to adapt between several users, as well as allowing the system to develop over time. Along with previous experience in developing neural classifiers, it seemed appropriate that a neural based classifier should be used. Neural networks can be extremely fast at classifying data in real time, and it was desired to reduce any possible lag in the system to an absolute minimum. Development time was also an issue, and previous experience with neural classifiers meant that the implementation of the system would be relatively quick. It was ultimately decided upon a Probabilistic Neural Network (PNN) as our classifier for a number of reasons, which are described below. The PNN gives an output of Bayesian probabilities, and each class has a probability associated with it. The class with the highest probability is taken to be the classification.

PNNs are a neural network similar to radial basis networks, combine some of the best attributes of statistical pattern recognition and feed-forward neural networks, as well as allowing new training patterns to be incorporated into a previously trained classifier relatively easily. For an online application this would seem like a major advantage. At the same time, PNNs are also very fast at learning, because the learning rule requires only a single pass through the training data. However, PNNs, this comes at the expense of larger memory requirements and slower classification of unknown patterns when compared to conventional

neural networks such as the multi-layer perceptron (MLP). However, given the small amount of generalisation that would be needed in this system, it seemed a worthy trade off.

PNNs are the neural network implementation of kernel discriminant analysis [14, 16], and are essentially a normalised radial basis function network where a hidden unit is centred at every training case. Because the training system is not iterative in the same way that a back-propagation network is, the training time is significantly faster.

A PNN consists of d input units, comprising the input layer. Each input unit is connected to each of the n pattern units with a weight, which are modifiable and are used to train the network. Each of the pattern units are in turn are connected to only one of the c category, or output, units. When training the PNN, each pattern \mathbf{x} of the training set is normalised:

$$\sum_{i=1}^d x_i^2 = 1 \quad (8)$$

The training patterns are then placed on the input units, and the weights linking the k^{th} input units to the k^{th} pattern unit are set: $\mathbf{w}_k = \mathbf{x}_k$. A single connection is then made from this pattern unit to the output unit corresponding to the known class of that pattern. The training of this type of network requires only one pass through the data.

This trained network can then be used for classification by placing a normalised test pattern \mathbf{x} on the input units, much the same as was done in training. The classification algorithm then calculates the inner product for each pattern unit, producing the net activation:

$$net_k = \mathbf{w}_k^t \mathbf{x}, \quad (9)$$

Each pattern unit then sends a non-linear function, f , of net_k to the connected output units. The output units then sum the functions passed to them from the connected pattern units. The non-linear function used has a parameter σ which determines the width of the Gaussian window.

$$f = e^{(net_k - 1) / \sigma^2} \quad (10)$$

The computed sum from the corresponding category unit gives the discriminant function, $g_j(\mathbf{x})$, and performing a maximum operation, $\max g_j(\mathbf{x})$, gives the output category,

Upon starting the BCI model in MatLab, the PNN is trained with a normalised data set consisting of RARX parameters of a known classification. This data consists of five examples of each classification, of which there are five, and allows the model to give some generalisation to unknown signals.

An expansion of this classification system could involve a committee of classifiers, where the averaged output of multiple classifiers is taken as the actual output and can increase the classification rate by a large amount [1].

3.4. Training Environments

Two distinct training environments were developed, so that the different learning methods could be compared. The first training environment is a very simple indication of left or right movement, and was written in C# to allow simple integration into the rest of the system – a graphical bar moves from left to right on the screen based upon the classification of the EEG signal. The user is given an indication of which signal they should attempt to produce, and a

coloured indicator shows whether they were successful or not – green is showed for a successful attempt, and red for an unsuccessful attempt.

The second training environment involves an information-rich, 3D PacMan game (Fig 6). An open-source version of the game was used, and modified for the needs of the system. This involved developing new maps, adding more dynamic control to allow different types of feedback to occur, and developing a Java DDE server to enable communication with MatLab, allowing movement control commands to be sent. The Java engine uses the Open Graphics Library, and as such can utilise the 3D hardware acceleration provided by the computer's graphics card. This meant that more CPU time could be devoted to signal processing and classification. As stated previously, the first hardware setup did not provide enough computing power, and this was partly due to the low powered graphics card used.



Figure 8. The 3D PacMan game, written in Java and using the OpenGL graphics acceleration library.

The user can play through this PacMan game by navigating a small map, collecting objects and avoiding 'ghosts'. Different map designs were developed, but it was discovered that narrow, corridor based maps offered the best option. Wider, more open maps tended to cause the user to lose control of their character as they tried to move about.

The 3D environment also provides more intense visual and aural feedback to the user – including sharp, noisy sounds for feedback showing an incorrect response, and calm, soft, music for feedback showing a positive response. Visually, incorrect feedback responses showed images of fire, and correct feedback response showed images of meadows. It is hoped that a measurable difference between the training times of the users who use the 2D environment and those that use the information-rich 3D environment.

One of the key issues involved interfacing MatLab and the external applications. The original approach involved using Windows API calls within a MatLab function to generate fake keystrokes. The applications listened for keystrokes, and would take that input accordingly. For example, a classification of imaginary left hand movement would generate a keystroke of the left cursor key and move the bars/PacMan accordingly. However,

incompatibilities between MatLab and the keystroke DLL meant that an alternative solution had to be found. MatLab includes native support for the Dynamic Data Exchange protocol, which is relatively easy to implement in other languages. As such, a Java based DDE server was written to allow the PacMan game to listen to commands directly from MatLab, without having to go through a DLL.

4. EEG Data Sampling

4.1 Mental Tasks and Data Recording

In order to allow identification of mental tasks both in online and offline systems, five different mental tasks were recorded offline for analysis and training of the classification system. Each of these mental states can then be used as a control signal. For each task, the user was recorded five times, and each data set was twenty seconds in length. The EEG data was recorded at 128Hz, and saved direct to disk.

Task 1: Baseline. The baseline EEG signal was used as a reference to compare the other EEG signals. The user was placed in a completely relaxed state with their eyes closed. Once the user had relaxed for several minutes, the data was recorded.

Task 2: "Blackboard" Counting. The "Blackboard" Counting task involves the user imagining drawing sequential numbers on a blackboard, erasing the number each time. The user practiced this for several minutes before being recorded.

Task 3: Non-Trivial Mental Arithmetic. The third task involved the user's EEG signals being recorded whilst involved in some non-trivial mental arithmetic task. The task had to be non-trivial to ensure the user was performing the arithmetic, rather than reciting "times tables". The user was given a large multiplication to perform (e.g. 72×93) whilst recording was taking place. It's worth noting that whether the user correctly answers the multiplication question is irrelevant, as it is the mental activity that is being examined.

Task 4: 3D Rotation. This task involved the user visually the rotation of a 3-dimensional object within their mind.

Task 5: Imaginary motor movement. This task involved the user imaging moving their left and right hands separately. This task is a classic BCI cognitive activity, but is different from the other tasks. The signal is unlikely to be stationary, as the other tasks will be, due to the fluctuation caused by the event-related desynchronisation in the motor cortex.

5. Results

Due to the modular nature of this system, each component was tested thoroughly before being integrated into the complete system. This enabled a greater understanding of each component, and the role it plays in the system. It also allowed us to debug each component thoroughly, and work on optimisation of each component.

5.1. Testing the EEG Machine

The EEG machine was originally tested by examining the pen/graph output from the rolling graph of the EEG machine. This allowed us to check out electrode placements for the next

part of testing, which involved developing a simple classification system originally designed in Simulink, but later rewritten using MatLab's Data Acquisition Toolbox. This involved a threshold function, to examine the output from the EEG machine. Electrodes were placed on the forehead, one above each eye, as well as a reference electrode placed on the earlobe (Figure 9). It was discovered that stationary movement of the eye lids produced very little output from the EEG machine, but a blink from one eye lid produced a recognisable pulse that peaked at around 2V. The classifier involved a basic comparator, and activated an output when the signal from the EEG machine went above 1.5V. This allowed the quick development of a multi-channel classifier that allowed the control of our external applications via left and right eye blinking, and demonstrated a complete system.

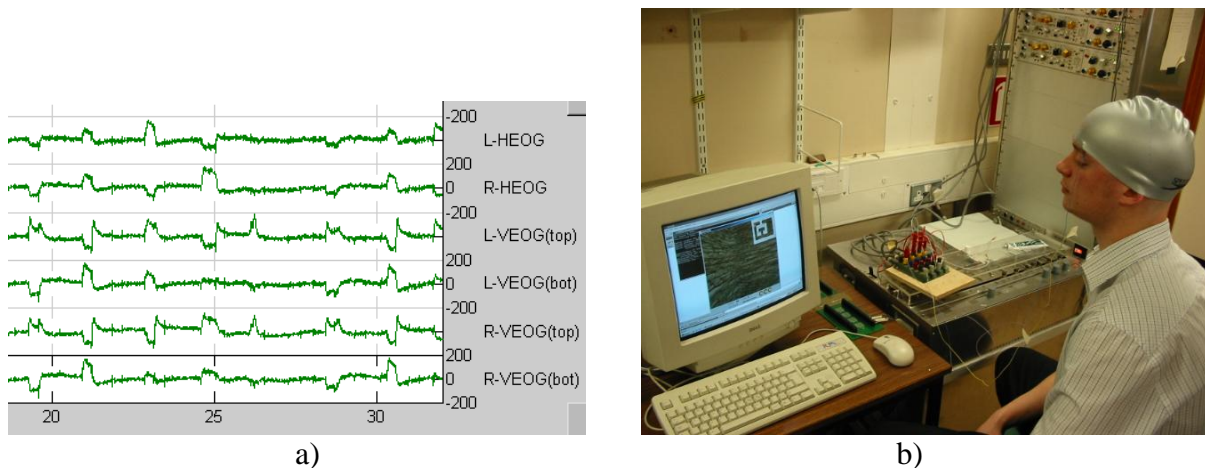


Figure 9. The “blinker game”. A) An example of eye blink data gathered, and b) our test user playing Pacman 3D with eyeblinks to control left and right movement.

5.2. Testing the RARX Feature Extraction

Once the development of the RARX modelling code was complete, it was tested using a signal created from random, but stable, AR coefficients. This was then compared to the modelling systems native to MatLab, and gave very promising results.

5.3. Testing the PNN Feature Classification

The PNN was designed to allow different data types to be classified, and this allowed testing with samples other than EEG feature data. Simulated surface acoustic wave sensor data was used to test the neural network, and the network gave a misclassification rate of 3.67%.

5.4. Testing the training environments

The testing of the training environments proved to be relatively simple, as we had allowed input to come from both DDE and the keyboard. This meant that development could be continued on computers without MatLab, and tested independently. Once the DDE server was written, basic MatLab scripts were run to ensure that successful connections and instruction poking could be achieved.

Upon completion of the EEG blinking model, this was used to further test the integration of the training environments in a real-time environment, and thus “close the loop”.

5.5. Testing the extraction and classification on EEG data

To test the extraction and classification systems, an offline system was developed. This system calculated the ARX parameters for each sample over the time of the data. These parameters were then used to train the PNN. Three sets out of the five recorded, for all five of the mental tasks defined within the project. The extraction and classification system were then tested using the other two data sets from the collected EEG data. Each of the two extra data sets from each of the mental tasks were concatenated together to produce 200 seconds of untrained test data. This meant that the PNN had to generalise, and classify data it had not previously seen. The results were encouraging, and for certain tasks – the imaginary motor movement of the left and right hands – the classification was > 90% successful (Figure 10).

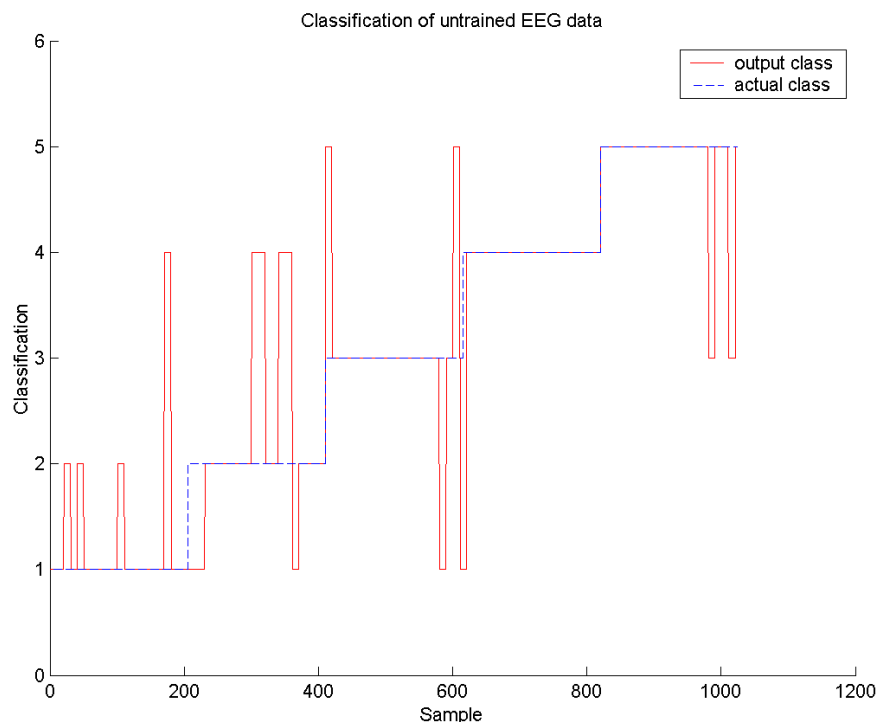


Figure 10. Graph of classification results showing the classification of untrained EEG data with the test user performing specific mental tasks.

It is worthy to note that the first class in the graph is the baseline task, where the user is in a relaxed state. This proved to have less successful classification than all of the other tasks, and this is believed to be because the baseline data recorded actually exists in all the data. Due to the training environment, the test user was not able to achieve a completely relaxed state, and background brain activity reacting to sounds, movements, and other activity may have led to these inaccurate classifications.

A technique is currently in development to remove this problem by performing a Fast Fourier Transform (FFT) on the EEG signal and subtracting the baseline FFT. This will remove all of the frequency features that exist in both baseline and non-baseline data, leaving only the frequency features specific to the mental task being performed. A reverse FFT is then performed on the result, and classified as normal.

Another approach later developed for the system to improve classification was modal smoothing. Since the classification was only occurring every half second, one

misclassification at that time would not necessarily be indicative of the actual mental task being performed. By adding in a modal smoothing function for every classification, the previous half second of data can be classified and the most common classification is used as the task identifier. The performance loss from including this is negligible, but the classification gain is considerable, and in some cases improved the classification rate from 75% to 95%+.

5.6. Testing the system

Once all the system components had been tested individually, they were integrated together. Once this had been completed, and compatible data types were being passed from one component to another, and any component connection issues resolved, the complete system testing could begin. This involves the training of different users in the 2D training system, and the 3D training system. Several sessions are needed to train the individuals, and this will allow us to gain feedback on the system and make any necessary improvements.

Unfortunately, due to sampling errors in the real-time system, it has not yet been possible to run a real-time BCI system. This is due to samples being dropped in the system, as the callback function used to gather the data has a lower priority than the extraction and classification functions. This is being remedied by using a higher priority callback function within MatLab, called the `SamplesAcquiredFcn`, rather than the `TimerFcn` callback.

6. Future Work

This project still has a lot of potential for further work, and is by no means complete. Work is still ongoing in developing the real-time system. Although the coding of this is complete, sampling issues mean that the data is not being presented to the extraction and classification functions in the same manner that the offline data was.

Further work would also include developing a system where the signal analysis and classification would run on a different computer to the one the training environments are running on. This would allow for an increase in the computing power available to the training environment, and greater development of a more immersive environment – perhaps with some augmented reality using 3D glasses.

More thorough user testing will be carried out to enable proper analysis of the system, and the differences in learning rates between individuals using the 2D training system and those using the 3D information-rich environment.

The modular way in which the BCI system described in this paper has been developed will allow other extraction and classification mechanisms to be used with relative ease, and future work may involve comparative analysis of different feature extraction and classification systems. Future classification systems would likely involve a committee of classifiers, although the impact to performance in a real-time system would need to be examined to ensure that the system would still operate effectively.

7. Conclusion

This paper has demonstrated the background and uses of a brain computer interface, along with a successful design and implementation. It can be seen that there are many applications for BCI systems, especially in the medical and rehabilitation field where many lives could be improved through the use of prosthetics despite complete paralysis.

It can also be seen that there are many ways in which a BCI could be implemented, and the development can continue to test these approaches.

8. Acknowledgements

This project was developed jointly with Alex Green, also of the University of Reading. The original inspiration, along with much support and advice came from Dr. Slawomir Nasuto, Mr. Julian Brooker, and Miss Nicoletta Nicolaou, all from the University of Reading.

9. References

- [1] William D. Penny, Stephen J. Roberts, Eleanor A. Curran and Maria J. Strokes. EEG-based communication: a pattern recognition approach.
- [2] Raquel Prado, Mike West and Andrew D. Krystal. Multi-channel EEG analysis via dynamic regression models with time-varying lag/lead structure.
- [3] Christoph Guger, Werner Harkan, Carin Hertaes, Gert Pfurtscheller. Prosthetic Control by an EEG based Brain Computer Interface (BCI)
- [4] S.J.Roberts, W. Penny and I. Rezek. Temporal and spatial complexity measures for EEG-based brain-computer interfacing. *Medical and Biological Engineering and Computing*. 37(1):93-99, 199.
- [5] S. Balakrishnama, A. Ganapathiraju. Linear Discriminant Analysis – A Brief Tutorial.
- [6] Jose del R. Millan, Josep Mourino, Fabio Babiloni, Febo Cincotti, Markus Varsta, Jukka Heikkonen. Local Neural Classifier for EEG-based Recognition of Mental Tasks. *IEEE-INNS-ENNS International Joint Conference on Neural Networks, July 24-27, Como, Italy*.
- [7] H. Lusted and R. Knapp, "Controlling computers with neural signals," in *Scientific American*, Scientific American, October 1996.
<http://www.sciam.com/1096issue/1096lusted.html>.
- [8] J. Walpaw, D. McFarland, and T. Vaughan, "Brain-computer interface research at the wadsworth center," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, June 2000.
- [9] N. Birbaumer, A. K" ubler, N. Ghanayim, T. Hinterberger, J. Perelmouter, J. Kaiser, I. Iversen, B. Kotchoubey, N. Neumann, and H. Flor, "The thought translation device (tt) for completely paralyzed patients," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, June 2000.
- [10] J. Walpaw, N. Birbaumer, W. Heetderks, D. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. Quatrano, C. Robinson, and T. Vaughan, "Brain-computer interface technology: A review of the first international meeting," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, June 2000.
- [11] S. Levine, J. Huggins, S. BeMent, R. Kushwaha, L. Schuh, M. Rohde, E. Passaro, D. Ross, K. Elisevich, and B. Smith, "A direct brain interface based on event-related potentials," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, June 2000.
- [12] S. Makeig, S. Enghoff, T.-P. Jung, and T. Sejnowski, "A natural basis for efficient brain-actuated control," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, June 2000.
- [13] E. Curran, P. Sykacek, M. Stokes, S. Roberts, W. Penny, Ingrid Johnsrude, Adrian M. Owen, "Cognitive tasks for driving a Brain Computer Interfacing System: a pilot study"
- [14] Specht, D.F. (1990) "Probabilistic neural networks," *Neural Networks*, 3, 110-118.

- [15] C. W. Anderson, E. A. Stolz, and S. Shamsunder, "Discriminating mental tasks using eeg represented by ar models," in *IEEE Engineering in Medicine and Biology Annual Conference*, IEEE, 9 1995.
- [16] Hand, D.J. (1982) Kernel discriminant analysis. Research Studies Press. Coomans, D. and Broeckart, I. (1986) Potential pattern recognition in chemical and medical decision making. Research Studies Press.