2010

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

USING 3D VIDEO GAME SCENARIOS AND ARTIFICAL NEURAL NETWORKS TO CLASSIFY BRAIN STATES FOR A BRAIN COMPUTER INTERFACE

Brent Maurice Benson

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems

University of North Carolina Wilmington

2010

Approved by

Advisory Committee

Devon Simmonds Ling He

Karl Ricanek, Jr
Chair

Accepted by

Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

A brain-computer interface (BCI) is a technology that allows the user to interact with a computer without relying on any overt physical activity, i.e. hand movement as with a mouse or finger movement with a keyboard. A BCI works by monitoring the user's brain signals, extracting the important features, using the features to classify their intent, and then providing feedback to a computer application. This study focuses on an Electroencephalograph (EEG) based BCI designed to classify two dimensional manipulation of a remote control for the Nintendo Wii™ game console (Wiimote). The long-term goal of this project is to allow users the ability to interact with the system without having to physically operate the Wii remote. Such a system would allow a physically impaired user the ability to interact with the system.

For this study subjects were asked to play a simple 3D video game using the Wiimote. First, subjects had to respond to stimuli by holding the wiimote with both hands and tilting it to the left, right, up or down. Next, subjects were instructed to imagine manipulating the Wiimote to the same stimuli while attempting to avoid any physical activity. During the game, EEG was recorded from 64 electrodes covering the subject's scalp. Linear regression techniques were used to extract the important components in the data. These components were then used as input features for an Artificial Neural Network to classify the user’s intentions.

The results of this experiment prove that EEG contains movement related classifiable information for both physical and imagined movements.

ACKNOWLEDGMENTS

With so many people to thank, I don't know where I should begin. Thanks to Dr. Ricanek, Dr. Kline, and Dr. Simmonds, Lloyd Smith, and Julian Keith for your patience and time. A special thanks to all of my subjects for participating in this experiment, punch and pie will be your reward! Bill and Paul, family dinners made this possible. Jason and Zach, movies and steak keep a brother sane. Last but not least, Nico and Niara, blue fur is the best kind.

DEDICATION

This thesis is dedicated to my Mom and Dad, thanks for your never ending support…

LIST OF TABLES

Table Page

LIST OF FIGURES

1. INTRODUCTION

1.1 Overview

In the popular film “The Matrix” the characters used a connection in the back of their head to control their virtual selves with only their thoughts. Science fiction aside there is a considerable disconnect in how fast we can relay our thoughts to a computer. The average human has a typing rate of less than 50 words per minute and a reading rate of less than 300 words per minute. The transmission of information from computer to computer is multiple times faster than human to computer. Advances in human computer interfacing, like speech recognition, are closing the communication gap but are not taking advantage of our full potential. A computer that has a direct link to our brain and can interpret our thoughts would eliminate any physiological activity and potentially let us communicate with a computer at the speed of thought! Although we are still far away from plugging ourselves into the computer, breakthroughs in cognitive neuroscience and computer science have brought us closer to controlling computers with our thoughts. This cutting edge technology has been titled “brain-computer interfacing”.

1.2 Motivation

A brain-computer interface, BCI, is a system that allows a user to control computer functions by using brain activity. The most important application of BCI technology is for the disabled. A BCI does not require any muscle activity so it could be operated by people with severe physical disabilities. An often cited example are those suffering from Amyotrophic Lateral Sclerosis, ALS, a neurological disease that gradually disables muscle control and

eventually leading to full paralysis. ALS does not affect brain cells or cognitive function, resulting in the 'locked-in' syndrome made famous by Stephen Hawkins, where the patient is aware yet trapped in their own body. A BCI could analyze their brain signals and use them to communicate directly with a computer [18][22][33].

The 3D video game control is another avenue where a BCI could prove beneficial. Video game controllers have advanced through out the years but companies like Nintendo, Sony, and Microsoft are always looking for new and innovative ways to control their popular video game consoles. New interface devices like Sony's "EyeToy" and Nintendo's "Wii Remote" are proof of the demand for interesting and inventive hardware to control modern video games. A BCI could prove a novel and efficient way to control the video games of the future [16].

1.3 Problem Description

There are three objectives in this thesis. The first objective is to create a BCI software environment and workflow to monitor, capture, and analyze brain signals. The second objective is to test the software environment by conducting a BCI experiment. The final objective is to classify the brain signals off-line using an Artificial Neural Networks.

1.3.1 BCI Software Environment

To simulate a fully functionally BCI three out of the four components necessary will be implemented. This study is focused on the *acquisition* and *classification* of brain signals for a BCI; an application based on a BCI is not pursued. The acquisition, feature extraction, and the classification components will be built to analyze and classify brain signals off-line.

1.3.2 BCI Experiment

An experiment will be conducted to test the BCI Software Environment. This experiment will involve eliciting time-locked voluntary physical moment EEG data and will attempt to elicit time-locked imagined movement EEG data.

1.4 Limitations

To conduct a BCI study there are many limitations to over come. The human brain is a complex organ that is not completely understood. Problems also arise when studying human subjects.

1.4.1 Electrophysiology

Right now many complex problems are involved with trying to create the perfect BCI. One problem is that our current understanding of brain signals and brain electrophysiology is still limited. It is also difficult to extract meaningful data from the brain because of the ongoing noisy electrical activity. To overcome these limitations the subject is required to complete the same physical or cognitive task over numerous trials [33].

The study will also be looking at a group of people as a whole. Since no two brains are the same, subjects will have different responses to the same stimuli. This could make it difficult to find correlations in the group.

1.4.2 Subject Attrition

Without any compensation it is difficult to get subjects to participate in experiments. It is even harder to get subject to return for multiple trials of the same experiment. The goal of this

study will be to test 10 subjects and have all of them return twice.

1.4.3 Subject Attention

In this study the subjects will be asked to complete the same task for 15 minute intervals. Repeating the same task for that long could prove difficult for some subjects. There is a possibility the subject might become bored or uninterested and stop actively participating in the study.

1.4.4 Large Data Set

Every time the subject is presented a stimulus on screen that 400ms before the stimulus and 600ms after will be considered a trial. The EEG data will be sampled at 512 hz over 64 electrodes. That means for every trial there are 32768 data samples for every trial. Feature extraction will prove vital to obtain only the important samples.

1.4.5 Time

Gathering EEG data is a time consuming activity. It can take over an hour to attach the electrodes to the subject and make sure they are reading properly. This up front time cost could make it difficult to obtain a large sample.

1.5 Hypothesis

I believe a subject playing a 3D video game will elicit brain states that can be classified by an Artificial Neural Network. The Artificial Neural Network will be able to identify if a player is attempting to move the game control (both physically and imaginarily) or holding it

still. Success will be measured if the Artificial Neural Network is able to classify the movement and non-movement states better than chance.

1.6 Hypothesis approach

Subjects will control an avatar in a video game using the Nintindo “Wii Remote” while the electrical potential on their scalp is being monitored by an electroencephalograph (EEG). A subject will *physically* manipulate the “Wii Remote” or *imagine* manipulating the “Wii Remote” to respond to stimulus in the video game. An Artificial Neural Network will then be trained and tested classify all of the subjects’ reactions, real or imaginary, to the different stimuli. Success will be judged on the speed and accuracy of the neural network’s ability to classify the groups brain states.

2. LITERATURE REVIEW

2.1 The Human Brain

The brain, along with the spinal cord, makes up the central nervous system. The central nervous system delivers electrical signals from the brain to the rest of the body and receives electrical signals from the body back to the brain. The brain is a complex system and is still poorly understood. Questions like “How do we think” and “Where are memories stored?” are remain unanswered.

2.1.1 Neurons

The concept of the neuron was discovered by Ramon y Cajal in 1911. Ramon y Cajal is credited with figuring out that the brain was composed of individual autonomous cells instead of an interconnected web of cells. There are several types of neurons in the CNS and one of the most distinctive is the pyramidal cell.

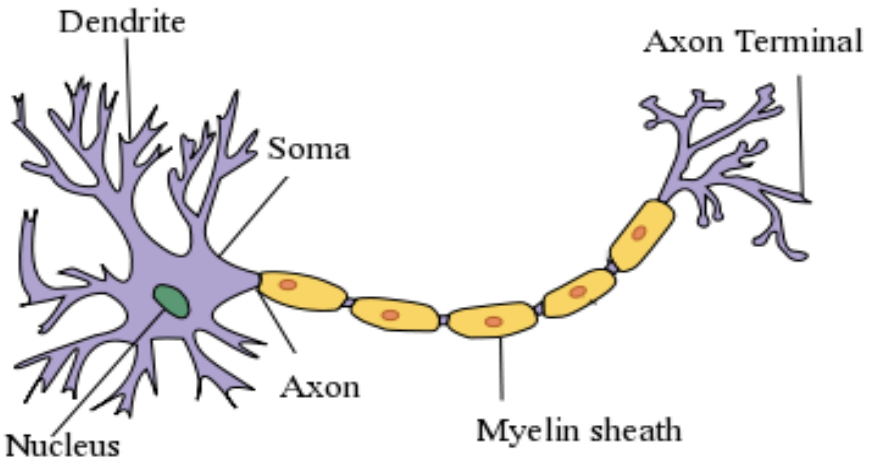


Figure 1. Illustration of a human axon [31].

2.1.2 Soma

Every neuron has a cell body called the soma. This is where the nucleus is located along with other organelles (ie. Mitochondria and endoplasmic reticulum). The soma coordinates the metabolic activity of the neuron. The dendrites, residing at the end of the dendritic tree act as inputs and the axons serve as outputs [12][31].

2.1.3 Dendrites

The dendritic tree is formed by the dendrites continually forking, making a finer and finer structure that can reach up to hundreds of branches. The dendrites that compose the dendritic tree receive signals from other neurons. An important property of the dendrites is their plasticity. Dendrites over time are able to sever and form connections with different nerve cells. This property is essential to the learning process[12][31].

2.1.4 Axon

The axon is long and slender structure that carries electrical impulses away from the

soma. Although axons are microscopic in diameter they can reach up to multiple feet in length. The sciatic nerve, for example, runs from the base of the spine to the big toe of each foot. Most axons are covered in a fatty layer called myelin, which increases electrical transmission. The end of the axon has a branch like structure called the axon terminal. The axon terminal is responsible for sending electrical signals into the gap junction between the terminals and the dendrites of the next neuron. This junction between the axons and dendrites is called the synapse [31].

2.1.5 Synapses

The cell walls of neurons are separated by small physical gaps called synapses. These allow electrical impulses to transmit between neurons. This electrical transmission happens via a chemical transmitter substance[31].

2.2 Neuronal Electrical Activity

Neurons are composed of fluids encased in very thin semi-permeable membranes. The permeability of the membranes allows for ions to pass in and out of the membrane. The fluid inside of a neuron has a high concentrations of potassium K+ and low concentrations of sodium Na+ and chloride Cl-. The outside of the cell is the opposite with high concentrations of Na+ and Cl- and low concentrations of K+. Electrical voltage is the result from the shifting of the ionic balance [12].

2.2.1 Action Potential

There are many reasons why the equilibrium in a neuron changes. When the cell is stimulated

the permeability of its membrane changes allowing ions to flow in and out of the cell. There is a sodium-potassium pump that maintains proper concentrations in the cell. High levels of ions overwhelm this pump causing a change in the resting electrical potential. If this change results in a voltage that exceeds –55 mV a special protein is released allowing for NA+ flowing freely into the cell for a fraction of a millisecond. This process is caused by depolarization and causes the potential to change up to +30 mV. After depolarization other proteins open up the cell membrane allowing K+ to flow out of the cell. This process is called repolarization. The action potential is a neuron's ability to produce a quick positive electrical charge. This electrical potential is what is monitored for a BCI [12].

2.3 EEG

Summated electrical activity generated from the brain can be recorded on the scalp. This recording is called the electroencephalogram (EEG). An EEG recorder uses electrodes placed on the scalp to measure brain signals. When using an EEG in conjunction with a non-invasive BCI the number of electrodes used to gather brain signal data is an important issue. As the number of electrodes increases, clarity of signal also increases, but operation complexity also increases. In the past, as few as two electrodes have been used, while the standard has been around 4-8 electrodes. Psychological EEG experiments sometimes use up to 256 electrode arrays.

2.3.1 EEG Electrodes

Electrodes are placed around the scalp using paste or gel and are usually held in place by a cap or a net. The gel aids in electrical conductivity allowing for a constant and more accurate reading.

Electrode locations and names are based on the International 10–20 system. The 10 – 20

comes from the spacing between electrodes. Fig. 2 is an illustration of the internationally

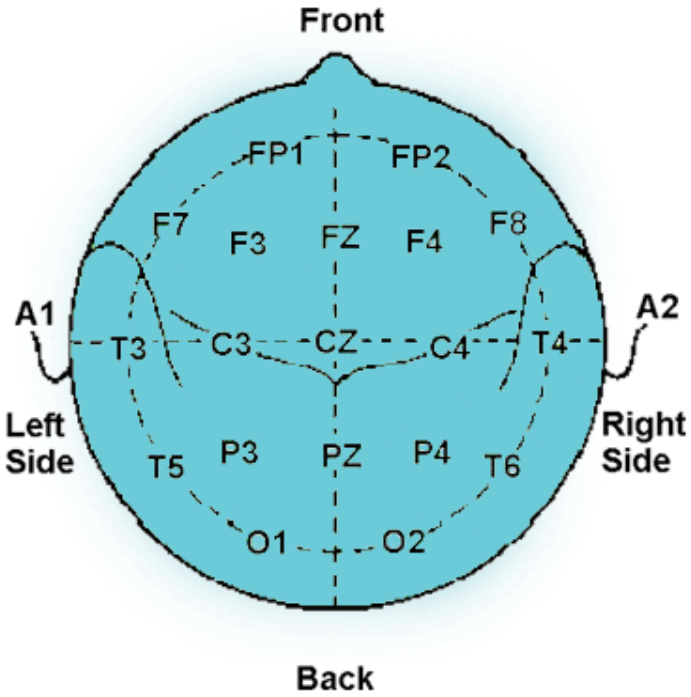


Figure 2. Illustration of internationally accepted locations for EEG electrodes [23].

accepted pattern for electrode placement where F - frontal, T - temporal, C - central, P - parietal, and O – occipital [12].

2.3.2 Number of Electrodes

Researchers have had success using different numbers of electrodes to read different types of ERPs. Average multi-channel EEG research involves anywhere from 32 to 256 electrodes. The goal in a BCI is to use the least amount of electrodes necessary to gather salient data. A BCI with too many electrodes becomes costly, cumbersome, and less feasible [25]. In early P300 ERP research only one electrode was necessary [11]. When reading the steady state visual-evoked response ERP researchers can gather a valid signal with only two electrodes [15].

2-8 electrodes have been capable in reading the μ-rhythm.

2.3.3 Filtering EEG Signals

Filtering the brain signal can potentially increase the speed of the calculations necessary for the classification component. Modern EEG electrodes are connected to an amplifier that enhances and filters the signal before it is sent to the computer. The amplifier can use a high-pass, a low-pass, and a notch filter to remove unwanted frequencies. The high-pass filter is used to remove DC-components, the low-pass filter removes high frequency noise, and the notch filter removes all frequencies outside of a specified range[12] [3]. Components are different pieces of the brain signal that can either be positive like ERPs or negative like eye blinks.

2.5 Event Related Potential (ERP)

After the signal from the brain is filtered, patterns can be found linking the brain signal to user intention. Oscillations in electrical potential will continuously occur between any two recording electrodes (ie. EEG) but the frequency and amplitude of these oscillations are influenced by different states of awareness. A person's EEG can be altered into different states during excitement, drowsiness, sleep, coma, anesthesia and various other activities. With direct or peripheral deterministic stimulations (electrical, optical, acoustical, etc) a person's EEG can be influenced. These induced changes in the brain's electrical activity are considered "event related potentials" (ERPs). ERPs are electrical responses to the presentation of some stimuli. An ERP is calculated by obtaining the average EEG from the raw EEG recorded on the scalp. A classifier must be built to look for these patterns and to distinguish between user intent [11][20].

An ERP is defined by three characteristics: the component, event proceeding component,

and event following the component. A component is a particular EEG response to certain stimuli that can be physiologically generated by the brain system or is created by a specific psychological process. An event proceeding component is brain activity that happens before the stimuli is reacted to or what happens in the subject's mind before she actually moves her hand. Event following components are the opposite and occurs after the stimuli is reacted to or after the subject moves her hand. An overview of ERPs can be found in Rug and Coles book "Electrophysiology of Mind: Event-Related Brain Potentials and Cognition" 1995.

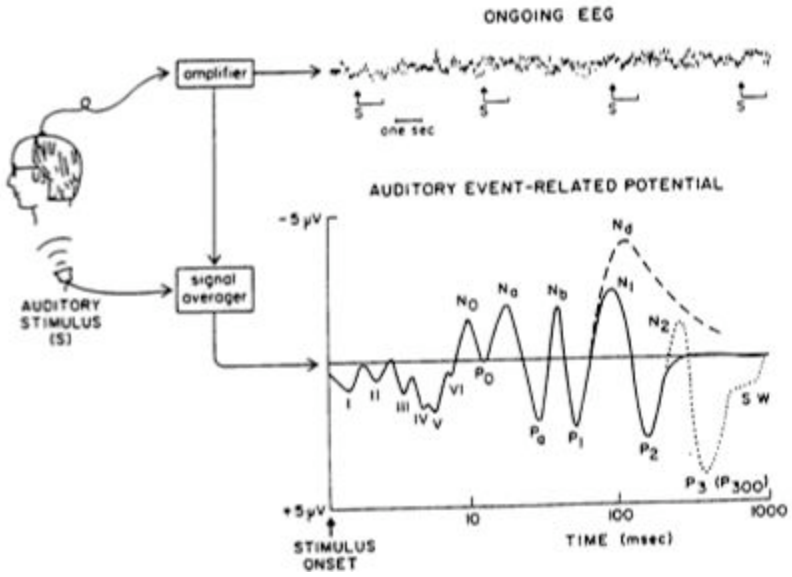


Figure 3. A diagram depicting different ERPs [20].

2.5.1 μ-rhythm

It has been found that motor functions such as hand movements are accompanied by event-related changes in the electroencephalogram. There is an 8-12 Hz (μ-rhythm) in the motor cortex that disappears during execution or planning of a movement. Studies have shown subjects are able to 'learn' how to manipulate this rhythm; thus, making it another event related potential for BCI use [23].

2.5.2 SSVEP

The Steady State Visually Evoked Potential (SSVEP) is like the P300 and the μ response in that a stimulus event will create a predictable EEG response. SSVEPs are acquired by the repetitive presentation of a visual stimulus and measuring gaze direction. When the 'repetitive presentation' changes so will the gaze direction creating a change in the subject's EEG response and evoking a SSVEP [26][15][21].

2.5.3 Brain States

Erol Basar, a neuroscientist working in Germany and Turkey, discusses how changes in technology can help us classify brain states instead of just specific ERPs. Computer hardware is fast enough now to run data analysis techniques like wavelet analysis as a filtering component in parallel with the other components to form a BCI. This increased computing power can be used to process the large amounts of data coming from the EEG. Now the signal coming from every EEG electrode can be calculated equally instead of compressing the signal. Compressing the signal, like taking the average of all the electrodes, helps people analyze the ERP components but analyzing the electrodes together is very difficult for a person. That is why different pattern recognition techniques like Principal Component Analysis, Support Vector Machines, and Artificial Neural Networks are also necessary to classify these separate brain states. The idea of classifying brain states is the perfect problem for interdisciplinary research between neuroscience and computer science [1].

2.6 Brain Computer Interface

A BCI allows a user to manipulate a software program without any physical gestures required. An efficient BCI system has four identifiable traits. The first trait is the ability to extract relevant information from the brain signals correlated to specific mental tasks or events. The second is to adapt and self-learn in ever-changing noisy brain signals. The third trait is to predict user intentions (such as movement planning) both quickly and reliably. The last trait is the BCI taking appropriate actions (ie. control a device, give neurofeedback to a user) [9]. The four traits are carried out by the four basic components BCI system [28].

2.6.1Components of a BCI

The first component is actually obtaining the subject's brain signals. There are many methods to gather the brain signals; each method has its own pros and cons which will be discussed later.

The second component is a method of feature extraction or filtering out the brain signals necessary for the BCI to work. Superfluous signals in the background EEG need to be filtered out for more efficient processing. Algorithms like Fast Fourier Transformation, autoregressive modeling, and wavelet analysis are used for feature analysis and detection.

The third component is a classification system. Once the subject's brain signals are filtered they must be categorized by the subject's intent. A classification system needs to categorize and identify the subject's intent quickly and accurately. Classification success has been achieved using Artificial Neural Networks, Adaptive Logic Networks, and many other types of pattern classification systems.

The fourth and final component of a BCI is the application. This application translates the subject's intent (classified in the third component) into an action for instance moving a cursor on

a computer screen or directing the path of a wheelchair.

2.6.2 Brain Signal Acquisition

There are various aspects to consider when deciding what brain signal acquisition implementation to deploy: clarity of the signal, subject preparation time, danger to subjects, and long term effects on subjects all must be taken into account when building a BCI.

Brain signals can be acquired by inserting electrodes into the brain and monitoring individual neuron activity [14] but this is considered invasive. The advantages to an invasive BCI are very clear brain signals due to the fact that electrodes are implanted directly in the brain [14]. Multi-electrode arrays or sets of microelectrodes are used to record action potentials of single neurons in the brain [10]. The disadvantages to invasive BCI besides the obvious dangers of a complicated surgery, risk of brain damage, and infection are that electrodes can move relative to the individual neurons that are being monitored and scar tissue forms therefore decreasing the signal over time [14][10].

Partially- invasive BCI measures the subject electrocorticogram (ECoG) with nodes placed either subdurally (between the meninges and the brain). A subject's electrocorticographic (ECoG) activity is the electricity on the cerebral cortex [17]. The major disadvantage (besides the surgery) is the inability to read the firing rate of a single neuron. Like the EEG the ECoG measures the electrical field above the brain [17][10].

Lastly nodes can be placed on the scalp and a subject electroencephalographic activity (electricity on the scalp) can be monitored non-invasively [11]. Non-invasive BCI practices have a very significant advantage over the invasive and partially-invasive methods because it does not require surgery. Non-invasive BCI employs an electroencephalograph, with electrodes placed on

the scalp, to read the electrical reading off the scalp. The major disadvantage is that the signal isn't as clear as the other two methods and generally requires more training [14][17][10].

2.6.3 Filtering System

Digital filtering can be done in software to algorithmically normalize the data to focus on components of interest. Fast Fourier Transform and autoregressive algorithms have been employed with success in previous BCIs [11][12]. Some other BCI systems don't employ a software filter online for speed [19].

2.6.4 Classification System

Many methods have been used to attempt to classify brain signals. Principal Component Analysis, Support Vector Machines, Hidden Markov models, and many other methods have been used to classify brain signals. The classification systems used in this study are Artificial Neural Networks.

Artificial Neural Network's (ANNs) have been applied to a variety of problems from medicine to physics. An ANN can approximate a continuous function in an N-dimensional space. An ANN achieves hyper-dimensional function approximation by using feed-forward piecewise linear functions. The multi-layer perception with the back propagation learning law is a popular ANN model for classification and function approximation. Multi-layer perceptron ANNs are composed of several simple interconnected neurons. One side of neurons is the input layer, the opposite side is the output layer, and the neurons inbetween are considered hidden nodes. The connection coming to the neuron is a synapse and every synapse has a weight. Each

neuron's output is generally related to the state of the neuron and affects surrounding neurons. See figure 4 for a visual representation.

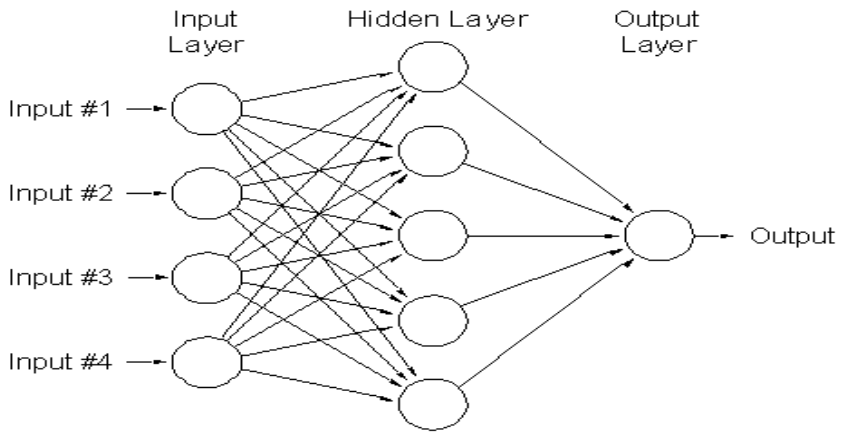


Figure 4. Diagram of a feed forward neural network.

2.6.5 Application Component

Once the brain signal is acquired, filtered, and classified, the next step is using the signal in an application to potentially benefit the subject. This work will use Blender 3D, which is an open source 3D modeling, animation, compositing, and video game creation suite. Blender lends itself to rapid application development by being open source and fully integrated with the computer programming language Python. Python is an interpretive language which means it does not need to be compiled. Python contains libraries to communicate with other popular programming languages like C, C++, and Java. By being integrated with Python the classification component can be written in almost any popular programming language and will still be able to interact with Blender.

2.6.2 Training a BCI

Right now it is unthinkable for a BCI system to interpret a subject's brain signals without a significant training period. An amount of adapting has to be done before a BCI system can properly interpret a subject's brain signal. This adopting can happen in two different ways: either the subject can adapt to better manipulate the system or the software is designed to adapt to a specific user.

Less complex systems are needed if the user is expected to learn to manipulate the system. The user learns by performing a task in a static system through a biofeedback process. The downside to this process is it can require an extensive amount of user training time. Complex software is required to create a BCI that will adapt to a specific user. Adaptive self-learning software would be used as the classification component. The idea is that the software burdens more of the learning task. These systems are usually faster and easier to use. Their downside is that the system is more complex and more difficult to develop.

2.6.3 BCI Implementations

BCIs have been implemented successfully in many different applications. Using behavioral conditioning and electrodes implanted in their brains, mice have been trained to think of pressing a lever (that is no longer there) to receive food (see figure 5) [8]. Various monkeys have also been trained to operate mechanical limbs using a BCI as illustrated in figure 6 [7]. There have also been positive results with various BCI applications in humans.

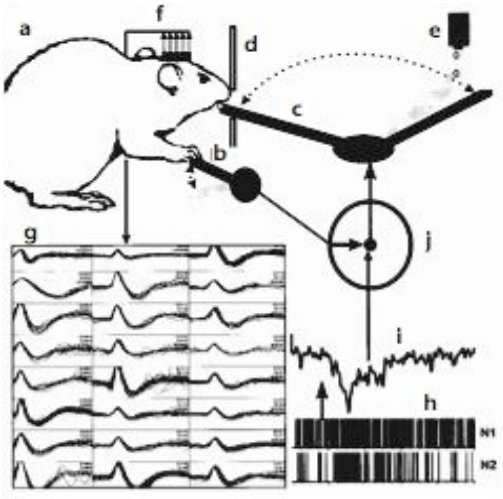


Figure 5. Probes are inserted directly into the brain of the mouse [8].

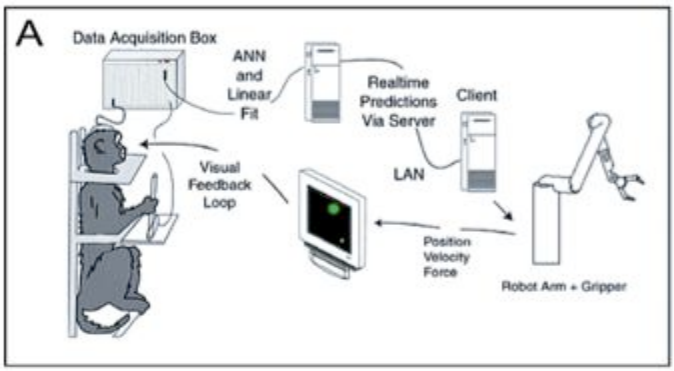


Figure 6. Using a BCI to remotely control a robotic arm [7].

Scientists have had recent success implanting electrodes directly into a subject's brain, giving the subject limited use of a computer application. Subjects without any motor function or 'locked in' subjects have the response from individual neurons monitored by the implanted electrodes. In the future, scientists hope these subjects will be able to control neural prosthetics using this invasive BCI [14].

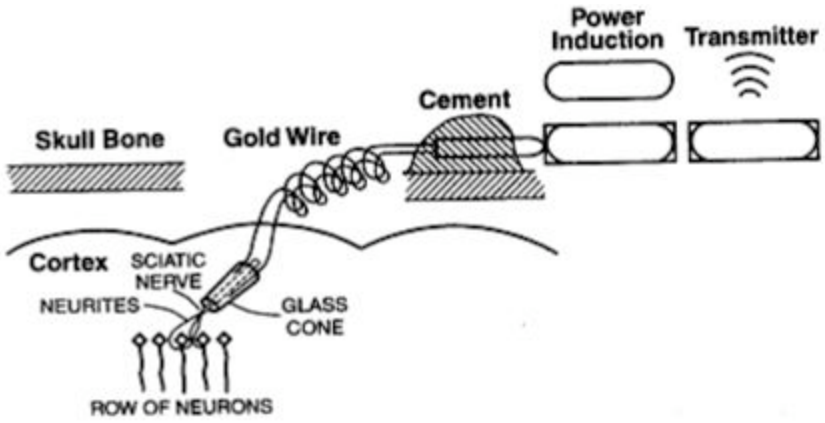


Figure 7. Diagram of a typical invasive BCI for a human[14].

Another successful method is implanting the electrodes between the brain and scalp or a partially invasive BCI. The subject's electrocorticographic signal which is electrical signals between the brain and the skull are monitored by a grid of electrodes placed on the skull or top of the brain. By imagining different physical motions like opening and closing right hand or saying the word 'move' or protruding the tongue, subjects were able to control a one-dimensional cursor [17].

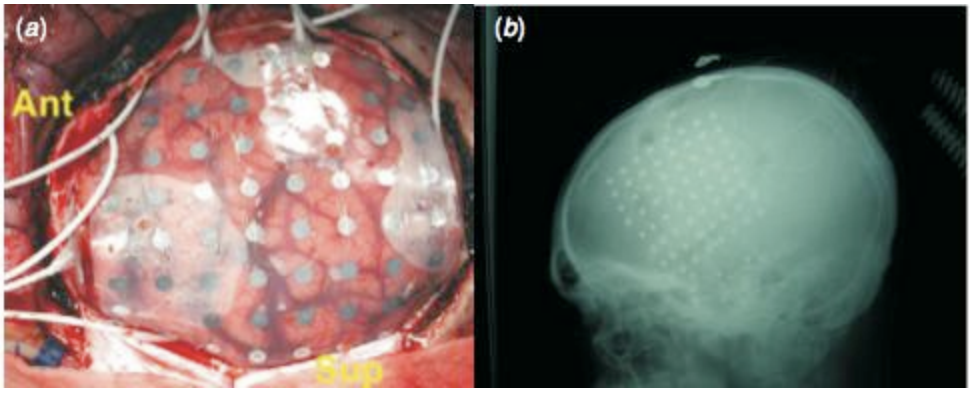


Figure 8. Example of a partially invasive BCI [17]

Using an electroencephalograph (EEG) to interface with a computer can also be used to create a viable BCI. There have been various BCI methods employed using an EEG. One of the first and most successful BCIs is Farwell and Donchin's P300 Speller. Farwell and Donchin

implemented a spelling program that works by continuously presenting the subject a 6 by 6 matrix of letters, numbers, and symbols as is demonstrated in fig. 9. An "oddball paradigm" is created by frequently and randomly highlighting either a row or column. The subject communicates a letter of desire by focusing attention on the cell containing the character. When the cell with the desired character becomes highlighted and a P300 response is provoked. Novice users of the system are able to use the P300 Speller after only a few iterations of training [11].

Figure 9. The type of display matrix used for the P300 Speller application

The Tubingen Thought Translation Device uses a subject's manipulation of the slow cortical potentials to select letters or words from a support program [4]. The Graz BCI uses differences in μ-rhythms or brain signals originating in the sensory-motor cortex to provide a control option in one dimension [24]. The Wadsworth BCI uses the μ-rhythm and the β-rhythm both from the sensory motor cortex to allow a subject to move a ball to a target on the screen [34]. There has also been work done trying to classify more abstract thought or higher level cognitive processing like asking people to imagine moving a video game controller [25].

3. Artificial Neural Networks

3.1 History of Artificial Neural Networks

The study of Artificial Neural Networks (ANN) and Artificial intelligence stems from the disciplines of Neurophysiology and Psychology. Breakthroughs in the mid 1800s gave rise to the ideas of a network of connected cells in the brain. In 1858 Joseph von Gerlach was able to use the stain carmine red, a bright stain obtained from some insects, to create some of the first images of interlocking neural cells [7]. It was first thought that these interlocking fibers were not individual cells but one network.

The idea that the nervous system was one set of interlocking fibers continued until the early 1900s. The first neural network was presented by Alexander Bain of the United Kingdom in his 1873 book entitled "Mind and Body. The Theories of Their Relation". Bain theorized that for memories to form the brain had to hardwire neurons together. This idea ends up playing a fundamental role in how ANNs function. Bain stated: "For every act of memory, every exercise of bodily aptitude, every habit recollection, train of ideas, there is a specific grouping, or coordination, of sensations and movements, by virtue of specific growths in cell junctions". Eventually even Bain came to doubt his own theories "The hypothesis was a legitimate one; but subsequent reflection led to the belief that the number of psychical elements, although run up to hundreds of thousands, was still inadequate".

In 1890 William James proposed that thoughts and bodily actions were the result of electrical potential flowing from parts of the brain having excess electrical charge to areas with less of a charge. James theorized this flow of electricity would depend on the intensity of the action. Between 1890 and 1910 it was proved that the electrical charge flowed down through

individual neurons. James explained that a neuron takes the electrical input from all of the neurons connected to it and if this input meets a certain threshold, the neuron fires its own electrical charge. The neuron firing is also known as an action potential. James' thoughts on the action potential would also show up in the basic workings of ANNs.

In 1938 N. Rashevsky proposed that the brain could be seen as a series of binary logic operations since action potentials could be viewed as binary values, 1 for a positive action potential, 0 for a negative potential. Rashevsky is able to prove how the 'Exclusive Or' operation would work but never proves any other operations. Working off Rashevsky's model, Warren McCulloch and Walter Pitts realized that working with multiple neurons in conjunction with thresholds could expand the number of operations the neurons could perform. McCulloch and Pitts proved the 'And' and 'Inclusive Or' operations. The ideas of thresholds allowed for multiple inputs but, at the same time, the operations ceased to be logical because they did not "define the basic set of state symmetry testers used as building blocks in larger circuits". Now the operations became parameter classifiers that classified their input patterns based upon the signal magnitude given by the summation of the binary inputs. These operations were considered to be calculating an analog value. This allowed for the operations to become adaptive using weights. The weight coefficient allows for neurons to have more or less influence on the output result. Modifying a parameter prohibits the input from being reconstructed and leads to ambiguity. This ambiguity is why ANNs are both lauded and condemned.

By 1963 brain researchers were spilt into two groups: neural network researchers and artificial intelligence researchers. Neural network researchers continued to try to define the parameter-based classifier networks while the artificial intelligence researchers attempted to let the network 'learn' the parameters. Since this split artificial intelligence researchers have used

neural networks in fields including speech synthesis, diagnostic problems, medicine, business and finance, robotic control, signal processing, computer vision and other pattern recognition problems.

3.2 Benefits of Artificial Neural Networks

As described in section 3.1 ANNs have been applied to many different problems offering a varied set of advantages. Generalization, non-linear processing, and adaptivity are a just a few of the benefits neural networks offer to problem solving. Non-linearity is an important benefit to ANNs.

Although an individual artificial neuron can be linear or nonlinear (because an ANN is made up of interconnected neurons it is inherently nonlinear). This characteristic allows ANNs to process nonlinear signals like speech patterns and forecast business data. For an ANN to process the data it must first learn the important characteristics.

For ANNs to function properly it must first be exposed to the data, this exposure is considered learning. After it has learned the data the ANN should be able to produce reasonable results for input it has not yet encountered. The ability to learn and generalize data is accomplished by making an input-output map of the data.

Input-output mapping is a type of learning called learning with a teacher or supervised learning (these ideas will be discussed further in section 3.3). This process involves adapting the synaptic weights of an ANN by subjecting it to a set of training samples where each input sample is provided with a desired response. The network attempts to adjust its synaptic weights to minimize the difference between the desired response and the actual response given by the network. This gives a nonparametric model (a model free of previous statistical assumptions) the

ability to form an input-output map of the problem data. This type of learning allows ANNs to adapt when presented with new data.

In ANNs adaptivity is the ability to adjust their synaptic weights to changes in the data. ANNs have the ability to retain their network structure while making minor changes in reaction to new data. ANNs can also be designed to update their synaptic weights in real time allowing them to constantly react to new data. ANNs can not only react to new data but also indicate how confident they are in their responses to the data.

An evidential response is the ability for an ANN not only to provide an output but also to respond regarding how confident it is in the decision. This response can help to identify ambiguous responses. Once the ANN has been trained, one can look at the contextual information to see how information is represented in the network.

Contextual information is represented by the structure and activation state of the ANN. Since every neuron in the network is potentially affected by the activity of the other neurons, the structure gives contextual information about how knowledge is represented in the network.

3.3 Learning

The goal of ANN learning is to take information from its environment and improve its performance. When an ANN is presented with new data it has the ability to readjust its weights and improve its performance. Mendel and McClaren defined ANN learning as:

> "…a process by which the free parameters of a neural network are adapted through a process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place."

The ability to learn and adapt to an environment is what make ANNs such useful tools. There are

different types of learning rules an ANN can be trained with.

3.4 Learning Rules

Every ANN must go through the learning process while using different learning rules to function effectively. Hayken describes the learning process happening in a sequence of three events: the ANN is stimulated by its environment, undergoes changes in free parameters as a result of the stimulation and, lastly, the ANN responds in a new way to the environment because of the changes that have occurred in its internal structure. As the ANN continues to repeat this process its goal is to improve performance. There are five basic learning rules when it comes to training ANNs. Each one takes a different approach to shaping the synaptic weights of an ANN during the learning process.

3.4.1 Error-Correction Learning

Error-correction learning is a type of supervised learning where the ANN is giving a set of input patterns with corresponding desired output patterns to learn from. Every iteration the ANN tries to decrease the difference between its actual output and the desired output pattern.

If we look at a single neuron the error signal is defined as:

$$e_k(n) = d_k(n) - y_k(n)$$

Let n be the time of the process involved in adjusting the synaptic weight and k represent the neuron. The actual output of the neuron is $y_k(n)$, the desired output of the neuron is $d_k(n)$ and the error signal is represented $e_k(n)$. The error signal stimulates a control mechanism that applies

corrective adjustments to the synaptic weight of neuron k. The purpose of the corrective adjustment is to bring the actual output of the neuron, $y_k(n)$, closer to the desired output, $d_k(n)$. This intention is actualized by minimizing a cost function, $\mathcal{E}(n)$. When defined in terms of the error signal, $e_k(n)$, the function is:

$$\mathcal{E}(n) = \tfrac{1}{2} e^2_k(n)$$

$\mathcal{E}(n)$ is the instantaneous value of the error energy. The synaptic weights of neuron k are adjusted step-by-step until the system reaches a steady state. This steady state is recognized when all of the weights are stabilized. After the weights are stabilized the learning process is terminated.

The learning process that deals with minimizing the cost function, $\mathcal{E}(n)$, is referred to as the delta rule. Let $w_{kj}(n)$ stand for the value of the synaptic weight w_{kj} of the neuron k excited by element $x_j(n)$ of the signal vector x(n) at time step n:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

η stands for a positive constant that will be regulates the rate of learning. As the network steps through the learning process this value is considered the learning-rate parameter. The delta rule can be stated as the adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse in question.

After adjusting the synaptic weight the updated value can be determined by

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

The values $w_{kj}(n)$ and $w_{kj}(n+1)$ are considered the old and new values of the synaptic weight.

3.4.2 Memory-Based Learning

In memory-based learning all or most of the past experiences are stored as properly classified input-output examples. In this method none of the data is generalized. The algorithm responds to novel inputs by comparing the inputs to already defined classes. If it is not an exact match the algorithm will attempt to retrieve and analyze the training data closest to the input.

3.4.3 Hebbian Learning

Hebbian Learning is considered associative learning because synaptic weights are changed when two neurons fire together. If two neurons on either side of the synapse are activated simultaneously the weight of that synapse weight is increased. If two neurons on either side of a synapse are activated asynchronously then that synapse weight is decreased.

3.4.4 Competitive Learning

In Competitive Learning the output neurons compete to become active. In Hebbian learning many neurons can be active at one time, but in a competitive learning scheme only a single output neuron is active at once. Individual neurons in this scheme learn to fire when certain trends are presented in the input patterns.

3.5 Learning Paradigms

When training ANNs all of the learning algorithms use the idea of the credit assignment problem. In essence, a credit is assigned to network configurations that perform well and blame is placed on configurations that do not perform well. There are two ways to conduct the credit assignment: supervised learning and unsupervised learning.

3.5.1 Supervised Learning

Supervised learning is also known as learning with a teacher. The teacher is an analogy for having information about the environment that is unknown to the ANN. This information comes in the form of input-output examples. For every input example the teacher is able to provide the proper response. The ANN readjusts its weights based on the input and the error in the output. The goal is for the ANN to achieve results close to what the teacher provides. This is how the teacher imparts knowledge to the ANN.

3.5.3 Unsupervised learning

In unsupervised learning there is no external teacher to oversee the learning process meaning there are no definitive examples to explain the environment. Within unsupervised learning there are two different paradigms, reinforcement learning and self-organized learning.

Reinforcement learning is conducted though continued interaction with the environment. An ANN is tasked with creating an input-output map to represent the environment. Once the ANN is tested a critic gives the ANN reinforcement by providing feedback on how well it performed. This type of feedback is called delayed reinforcement. This type of learning is difficult because there is no teacher to provide a response during the learning process and it is difficult for an ANN to adjust for mistakes because it doesn't know when they were made.

Self-organized learning means there is no teacher or critic to oversee the learning process. Instead a task-independent measure of how the ANN should represent the data is to optimize the free parameters of the network. The competitive learning rule is one way to perform self-organized learning. The input layer will receive data and the output layer of neurons will compete to activate. The task-independent measure will help determine which neuron gets to fire. A "winner take all" strategy can be employed so when one neuron is activated all the others will turn off.

3.6 Learning Tasks

ANNs have been trained to accomplish a variety of tasks in many diverse fields. Even though pattern association and recognition are two of the more popular learning tasks, there are many others.

Pattern association is the act of associating an input pattern with an output pattern. There are two types of pattern association: autoassoication and heteroassociation. In autoassociation an ANN is required to store a set of patterns then match an input pattern with one that is stored. In heteroassociation the ANN is given both of the input and output patterns then tries to match a given input pattern with one it has stored. Pattern association is not to be confused with pattern recognition

Pattern recognition is formally defined as the process where by a received signal is assigned to one of a prescribed number of classes. ANNs accomplish this takes by learning what input patterns are associated with certain classes. Then, when an ANN is given a novel input pattern, it estimates what class the pattern should be associated with. Pattern association and pattern recognition are different because pattern association tries to match an input signal with a

specific output signal and pattern recognition tries to best classify the input signal as a predefined set of classes, not necessarily a signal. Another learning task is the ability for an ANN to estimate a function.

Functional Approximation is when a general function is created to explain a set of data. When an ANN is given a series of input-outputs it is able to create a function relating the inputs to the outputs. Then when the ANN is given another input it can use the function to map what the output result should be. ANNs can also be trained to filter a noisy signal leaving only the important data.

The idea of filtering is extracting the important data out of a signal. Filtering can also be used to extract the important data out of a corrupted signal. Along with filtering data ANNs can also be taught to control complex systems.

The control task is the idea that given a set of inputs the ANN can estimate what action to take next. Using a series of feedback systems the ANN can be trained to monitor a system and control it in such a way to product optimal output. ANNs also have the ability to find out where there important signals reside in the data.

Beamforming is spatial filtering that distinguishes the spatial properties between the important part of the signal and noise. In other words, beamforming is locating the actual position of the important parts of the signal.

For this study an ANN will be required to perform a Pattern Classification learning task. Section 3.7 will discuss the different type of ANNs available to accomplish the task.

3.7 Popular Artificial Neural Networks

ANNs can use many different network configurations to accomplish a variety of tasks.

Some network configurations are better at performing certain tasks than others. Sections 3.7.1 – 3.7.5 will overview the different types of ANNs.

3.7.1 Single-Layer Perceptrons

A Single-Layer perceptron is a type of ANN consisting of just a single neuron with adjustable synaptic weights. Single-Layer perceptrons are also known as Rosenblatt perceptrons after he proposed the perceptron as the first model for supervised learning. Rosenblatt proved that if the patterns used to train the perceptron are drawn from two linearly separable classes, then the perceptron can create a hyper-plane between the two classes. A Single-Layer perceptron is only able to differentiate between two classes.

3.7.2 Multilayer Perceptron

A Multilayer Perceptron consists of an input layer, one or more hidden layers and an output layer. Multilayer Perceptrons are usually trained in a supervised manor using the error back-propagation algorithm based on the error-correction learning rule. Multilayer Perceptrons will be discussed in-depth in section 3.9.

3.7.3 Radial-Basis Functions Network

Radial-Basis functions networks (RBF) learn by finding a surface in multidmensional space that provides the best fit for the training data. After this surface is defined the network interpolates the input data by making it fit onto this space. RBFs consist of one input layer, one (and only one) hidden layer and a single output layer. The input layer takes in the signal and passes it to the hidden layer which performs a non-linear transform of the signal from the input

space to the hidden space. Lastly, the output layer supplies the linear response of the network depending on the network activation.

RBF networks are known to excel in complex pattern classification tasks. RBF networks solve the problem by transforming the problem into a non-linear high-dimensional space. Clover's theorem on the separability of patterns can be used to justify this transformation. Clover's theorem is based on the linear separability of patterns and, specifically, if the patterns are linearly separable then they can be much easier to classify. Let $\mathcal{H}$ denote a set of *N* pattern vectors $x_1, x_2, \ldots x_N$ each of which belongs to one of two classes, $\mathcal{H}_1$ and $\mathcal{H}_2$. If a surface exists in the family that separates the points in class $\mathcal{H}_1$ from those in class $\mathcal{H}_2$ a dichotomy is said to exist in respect to said surfaces. For each pattern $x \in \mathcal{H}$ we must define a vector made up of real-valued functions shown as:

$$\varphi(x) = [\, \varphi_1(x), \varphi_2(x), \ldots, \varphi m_1(x)]^T$$

If the pattern x is a vector in m0-dimensional input space then φ(x) maps the points from m0-dimensional space to a new space of dimension m_1. φ(x) can be referred to as a hidden function because it plays a role similar to a hidden node in a Multi-layer perceptron network.

A dichotomy $\{ \mathcal{H}_1, \mathcal{H}_2 \}$ of $\mathcal{H}$ is said to be φ-separable if there exists and m_1-dimensonal vector w that adheres to the following:

$$w^T \varphi(x) > 0, \quad x \in \mathcal{H}_1$$

$$w^T \varphi(x) < 0, \quad x \in \mathcal{H}_2$$

The separating surface in the input space is now defined as:

$$x: w^T \varphi(x)$$

A natural class of mappings can be obtained by using a linear combination of r-wise products of the pattern vector coordinates. The rational varieties of the r^{th}-order come from separating the surfaces by these natural class mappings. An r^{th} order product of entries x_i of x is considered a monomial. For an input space of dimensionality m0, the number of monomials is described as:

$$(m_0 - r)! / m_0!r!$$

Examples of the types of separating surfaces are hyperplanes for first-order verities, quadrices for second order varieties, and hyperspheres that are quadrices with certain linear constraints on the coefficients.

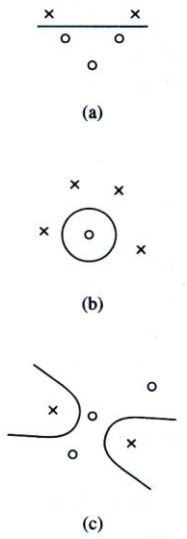


Figure 10. Three examples of φ-separable dichotomies of different sets of five points in two dimensions: (a) linearly separable dichotomy; (b) spherically separable dichotomy; (c) quadrically separable dichotomy.

3.7.4 Support Vector Machines

Support vector machines (SVM) are able to differentiate patterns by construing a hyperplane as a decision surface. In this decision surface the separation between positive and negative examples is maximized. An SVM can provide good generalization performance despite the fact it does not incorporate the problem-domain knowledge.

3.7.5 Committee Machines

Committee Machines (CM) use the idea of divide and conquer to accomplish their task. CMs work by dividing a complex task into a number of simple tasks and then combining the solutions. CMs use supervised learning and divide the learning task among ‘experts’ which in turn divide the input space into subspaces. A CMs topology consists of an input layer, a layer of elementary perceptrons (the experts), a layer consisting of a vote-taking layer, and the output layer.

3.8 Multi-Layered Perceptron

A multi-layered perceptron (MLP) has three main features that separate it from other ANNs. One feature is that every neuron in a MLP network has a nonlinear activation function. This is important because if the relationship between inputs to outputs was linear the MLP could be reduced to a Single-layer perceptron. The second feature is that an MLP contains an input layer, an output layer and one or more hidden layers. These hidden layers are what enable the network to learn complex tasks by extracting the important features from the input signal. The third feature is that MLPs have a degree of connectivity due to the amount of synapses between the input, hidden, and output layers. These three features are what give an MLP its processing power but inherent in these features are also the downsides to using an MLP [5][13].

The network uses the neurons in the hidden layer to derive complex synapse connections to represent the important features in the input data, but it is difficult for a human to find meanings in these connections. The presence of distributed yet highly connected non-linear neurons also makes an MLP difficult to train [5].

3.8.1 MLP Signals

An MLP uses two types of signals: a function signal and an error signal. A function signal represents the input signal that should, in turn, determine the output signal. The error signal is created in the output layer and is then propagated back through the network either reinforcing low-error neurons or prompting high-error neurons to readjust[5].

3.8.2 The Input, Hidden, and Output layers

In an input layer acts as a sensor organ to the MLP, allowing it to receive information from its environment in the form of the function signal. That signal is then passed to the hidden layer where the important features are extracted. The signal that the output layer produces is expressed in a continuous non-linear function of the input signal and the synaptic weights from the hidden layer. In addition to the output signal the MLP also produces an error signal. The error signal is a computational gradient of the error surface with respect to the synaptic weights of the neurons. This signal is then propagated back through the network to promote neuronal adjustment [5].

3.8.3 Back-propagation algorithm

While training a MLP the network uses the Back-propagation algorithm (BPA) to update the synaptic weights of the hidden layer. After a training iteration the output layer produces two signals: the output signal and the error signal. The error signal is then propagated back through the network with the goal of adjusting the synaptic weights to improve performance. The error signal is represented as:

$$ej(n) = dj(n) – yj(n)$$

where j is the neuron at the nth training sample. The instantaneous value of the error energy for each neuron is described as:

$$\frac{1}{2}e_j^2(n)$$

The instantaneous value of the total error energy is found by summing error energy values for all of the neurons in the output layer:

$$\epsilon(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n)$$

Let N stand for the total number of patterns contained in the training set. The average squared error energy is obtained by summing the instantaneous value of the energy over n and then normalizing with respect to N:

$$\epsilon_{av} = \frac{1}{N}\sum_{n=1}^{N} \epsilon(n)$$

The instantaneous error energy is a function of all the free parameters in the network. For any training set, $\varepsilon_{av\ is}$ cost function that serves as a measure of learning performance. The goal of the BPA is to minimize $\varepsilon_{av.}$ This is achieved by updating the synaptic weights pattern-by-pattern until one epoch (the entire training set) has been iterated through. The adjustments are made with respect to the error calculated for each pattern presented. ε_{av} is calculated in two steps that make the BPA: the forward and the backward calculation [13].

After a training epoch is presented both the forward and backward computations are calculated for every example. Let a traning example in an epoch be described as (x(n),d(n)) where the input vector x(n) is applied to the input nodes of the MLP and the desired response d(n) is presented to the output layper. The induced local fields and function signals are calculated by proceeding forward through the network layer by layer. The indeed local field $v_j^{(l)}(n)$ for neuron j in layer l is:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_o} w_{ji}^{(l)}(n) y_i^{(l-1)}(n)$$

where $y_i^{(l-1)}$ is the output signal from neuron I in the previous layer l-1 at iteration n and $w_{ji}^{(l)}(n)$ is the synaptic weight of neuron j in layer l fed from neuron I in layer l-1. If the neuron j is in the output layer where L is the depth of the network the final output is represented as:

$$y_j^{(L)} = o_j(n)$$

The error signal can now be computed:

$$e_j(n) = d_j(n) - o_j(n)$$

where $d_j(n)$ is the jth element in the in desired response vector d(n).

Now that we have an error signal we perform the backward computation and propagate the signal back through the network. First we must find the local gradient of the network. The output layer is represented as:

$$\delta_j^{(l)}(n) = e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n))$$

and the hidden layers are represented as:

$$\delta_j^{(l)}(n) = \varphi_j'(v_j^{(L)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n)$$

where φ_j' is the differentiation with respect to the argument. To adjust the synaptic weights of the network in layer l:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n)$$

where η is the learning rate parameter and α is the momentum constant. The forward and backward calculations are repeated for each sample in the training epoch until the stopping criteria is met [13].

3.9 Feature Extraction

Feature extraction, also known as dimensionality reduction, is the idea of reducing the number of samples passed to an algorithm by removing the unimportant and redundant data points. The chances for success increase by only presenting the algorithm the important. This helps to prevent the algorithm from processing the noise in data .Popular feature extraction methods include principal component analysis, Fourier transforms, autoregressive modeling, and regression analysis [13].

3.9.1 Regression analysis

Regression analysis is the attempt to model the relationship between a dependant variable and one or more independent variables. This model helps understand the change of the dependant variable when one or more independent variables change. The model of the

relationship is called the regression model. The regression model can be used to predict the dependant variable based on the independent variables. The regression model can also be used to extract the independent variables that best predict the dependant variable. This prediction element is what makes regression techniques good for feature extraction [13].

4 EXPERIMENT SOFTWARE DESIGN AND CONFIGURATION

4.1 Overview

The four components that make up a BCI will have to be implemented to conduct this experiment. Each component will either be developed specifically for this experiment or a customized configuration of existing software will be implemented.

4.2 Overview of the Acquisition System

To conduct this BCI experiment many systems must work in conjunction with each other. Figure 10 is an overview of all the systems necessary to conduct the experiment.

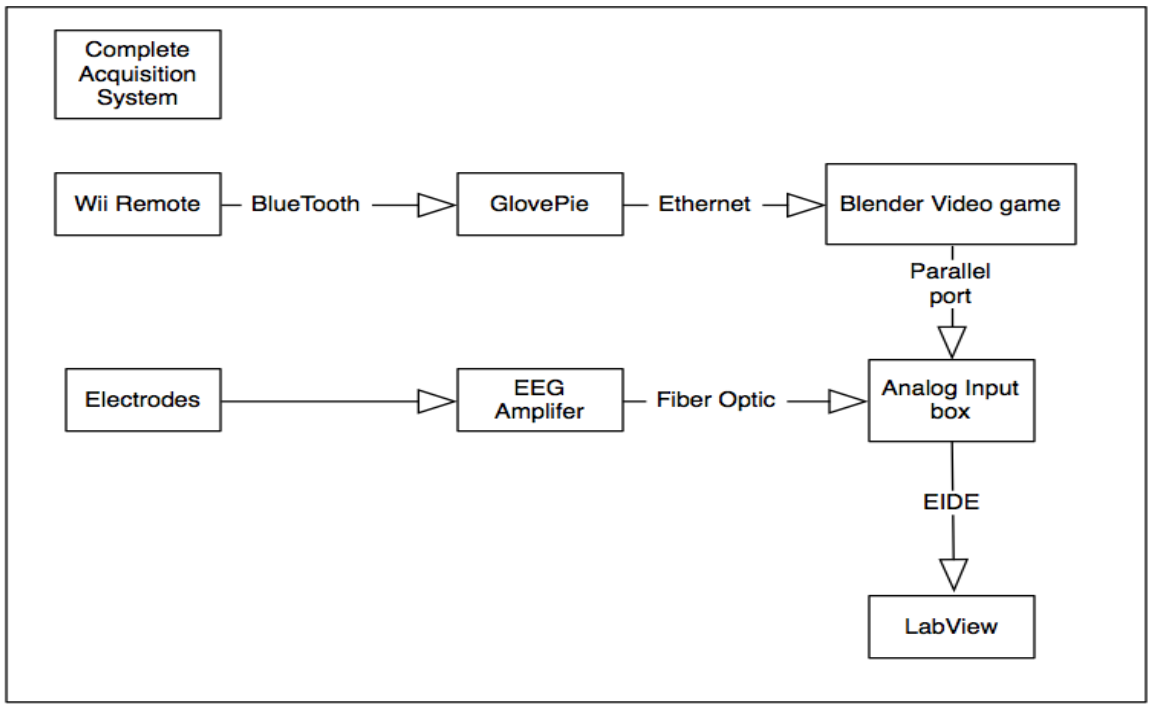


Figure 11. Diagram of the systems necessary to run the BCI experiment

Players will take control of the game using a Wii remote. The Wii remote will send

constant updates to the computer via BlueTooth. The BlueTooth signals are interpreted in GlovePie then sent over Ethernet to the video game created in Blender. Using the Python scripting language built into the Blender Game Engine, the signals sent over the Ethernet are interpreted into appropriate game actions. Every game action (a stimulus event or a subject's reaction to an event) is coded and sent to the EEG Analog input box via a parallel port. This is done so every action is marked real time on the EEG data.

At the same time the player is controlling the game the electrical potential off their scalp is being recorded by the EEG machine. The electrodes on their scalp transport the electrical signal on their scalp to the EEG amplifier (see section 2.3.3 for details on the EEG amplifier). The amplifier in turn sends these signals to the EEG Analog input box via a Fiber Optic cable. The EEG Analog input box takes the signals from the EEG amplifier and from the video game and sends all of these signals to the LabView application. Lab View allows a person to monitor the signals received from the EEG and video game in real time. All of these systems will be explained in full in section 4.3.

4.3 Experiment Acquisition Components

This experiment will use a non-invasive method to obtain the EEG from the individual subjects. A BioSemi Active 1 EEG will be used to collect the raw EEG data and timestamp the important stimulus and reaction events. The EEG consists of an amplifier and a battery. The battery is used because it is not safe to have a machine plugged into an electrical socket with direct links to a subject's head. The amplifier is used to enhance the minute electrical signals, measured in millivolts picked up by the electrodes on the scalp. The Active 1 EEG is able to use 64 electrodes to record from different locations on the scalp. This EEG uses a parallel port to

receive signals that are used to mark stimulus events in the EEG data. There is a proprietary graphical user interface named LabVIEW for this machine to control, monitor, and record the EEG signal. The application component of this experiment will be responsible for sending the stimulus and reaction events to the EEG.

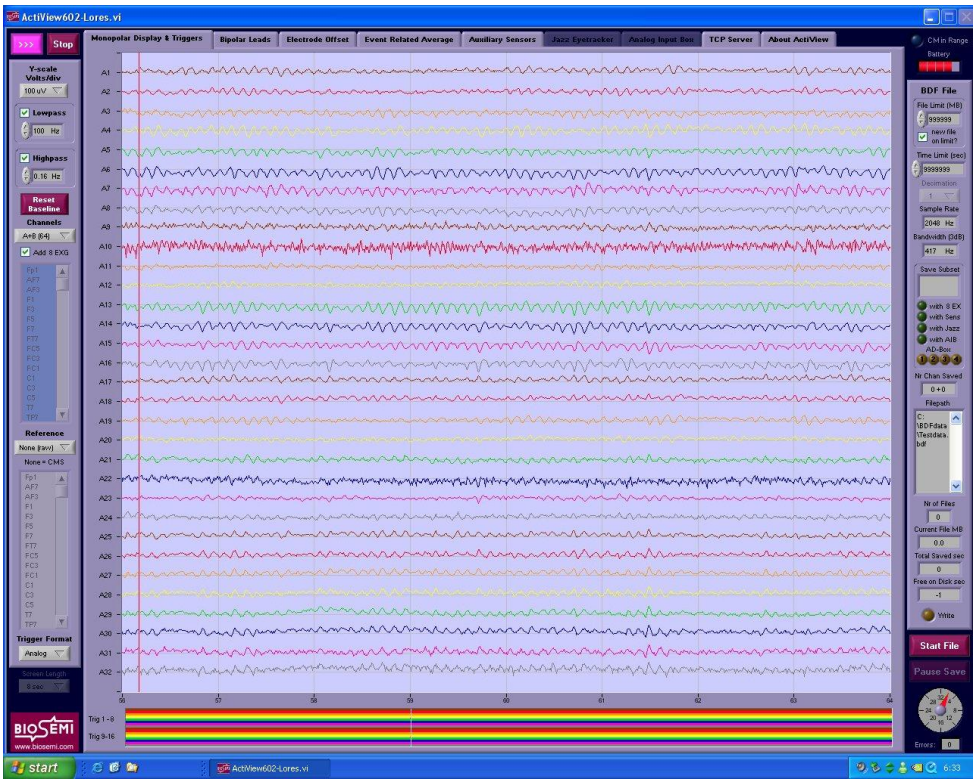


Figure 12. A screenshot of " labVIEW" the software used to capture, monitor, and save EEG Data

4.3.1 Experiment Video Game

A video game was created to elicit the brain states from the participating subjects. The game was created with open source program called Blender 3D. Blender uses Python scripting language to give developers more control over the game engine. Different scripts were created to control the timing of the game, displaying the stimulus on the screen, control how the subject interacts with the stimulus, and record these events. A script was also created to synchronize events in the game and the EEG signals. Each time a stimulus was presented or the subject

reacted to the stimulus the EEG data was marked with a code specifying the event.

In the game subjects took control of a spaceship on a mission to collect rings. Players will start by flying straight and each round the subject is presented a ring to the left, right, above, or below their spaceship or not presented a ring at all. A virtual Wii Remote is also displayed to instruct subjects on how to turn the controller. Figures 10 – 15 are screen shots of the games and where the rings appear.

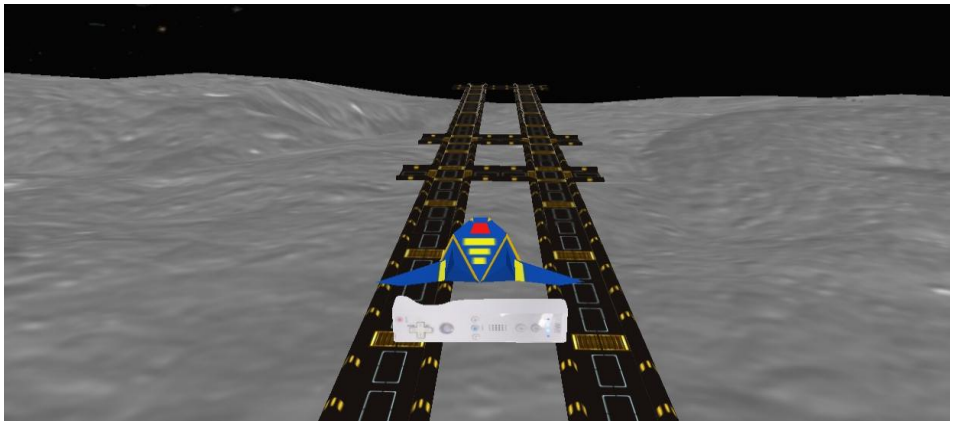


Figure 13. The spaceship flying at neutral with no ring presented

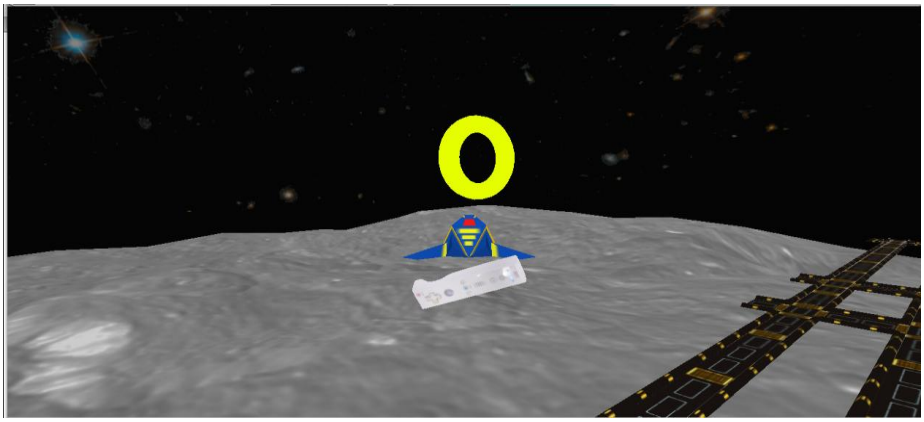


Figure 14. Ring presented to the left

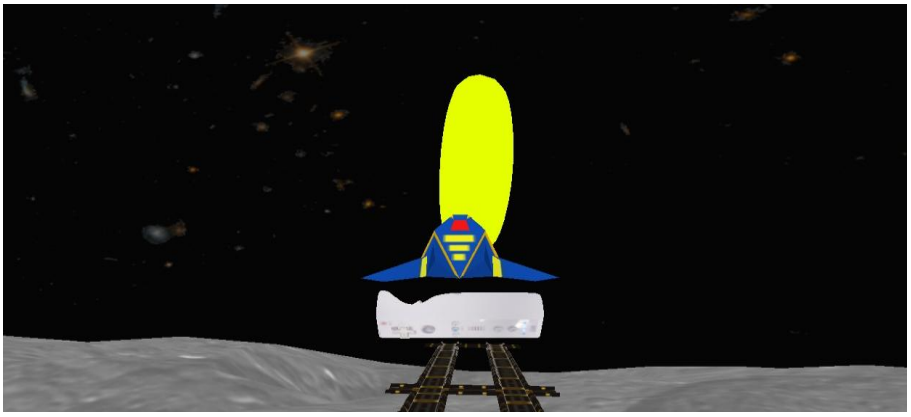


Figure 15. Ring presented to the above the spaceship

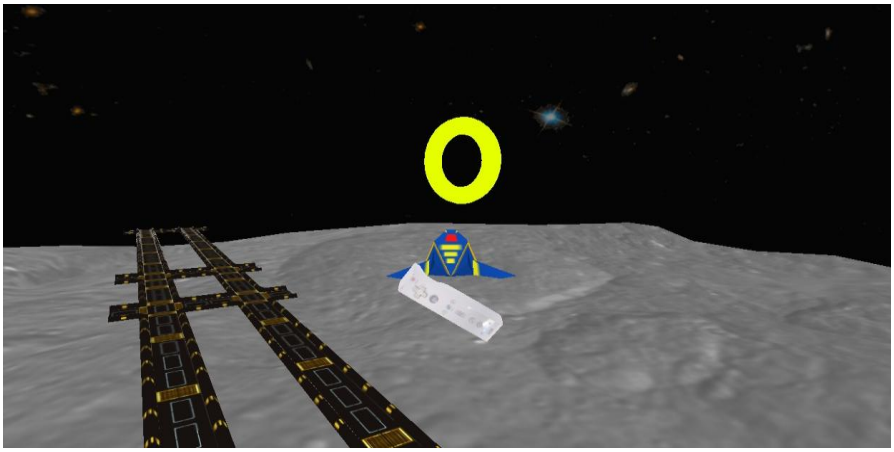


Figure 16. Ring presented to the right

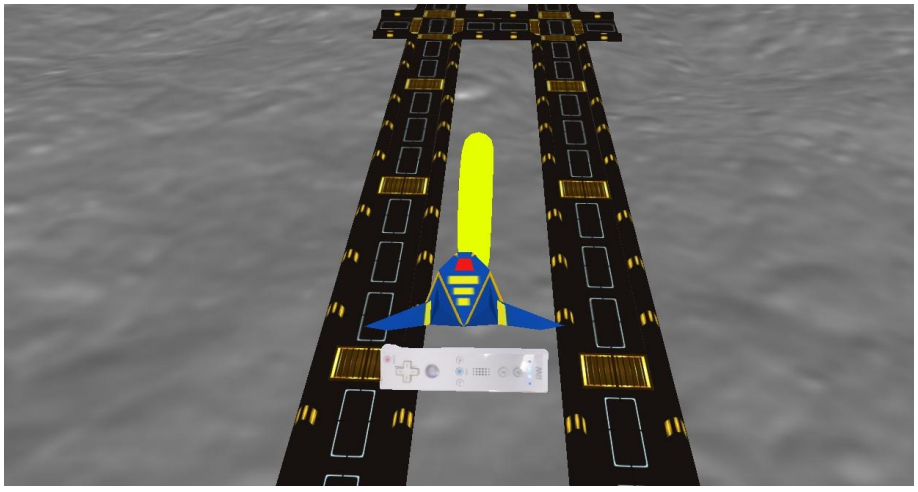


Figure 17. Ring presented beneath the spaceship

Subjects interacted with the game using a Nintendo Wii remote and manipulated it with both hands holding it like a steering wheel. For each stimulus displayed on the screen the subject was asked to perform a different gesture with the Wii remote.

Figure 18. Photograph of a subject playing the game created for the study

The Wii remote uses an accelerometer to calculate its pitch and yaw. This and other remote data is sent to the computer via Bluetooth© wireless interface. An open source library called GlovePIE was used to emulate the data received from the Wii remote and send it to the game over the Ethernet network. For the purpose of this study the controller must be moved at least 45 degrees to be considered a movement in the game. Figure 16 displays the 5 different gestures used for subjects to control the game.

Figure 19. Controller gestures for playing the game.

There were various reasons for choosing a Wii remote for this study. A user had to manipulate the Wii Remote with both hands therefore moving more muscles than pressing a button on a keyboard. It also gives a cutting edge user experience at a reasonable price. The GlovePIE makes it easy to use the Wii remote to control objects in a videogame made with Blender 3D.

4.4 Signal Analysis

Matlab was used to analyze the EEG data. Matlab stands for "Matrix Laboratory" and is a programming environment optimized for matrix math. There are two main reasons for choosing Matlab. One, there is an open source Matlab Toolbox for electrophysiological research called EEGlab. EEGlab offers a GUI and command functions to process and analyze EEG data. Two, the Neural Network Toolbox available for matlab offers a quick way to create, train, and test artificial neural networks.

The data from the EEG machine is saved as .BDF files. Once the file is read into EEGLab the data for each experiment is represented where the rows represent different electrodes and the columns represent the sample times.

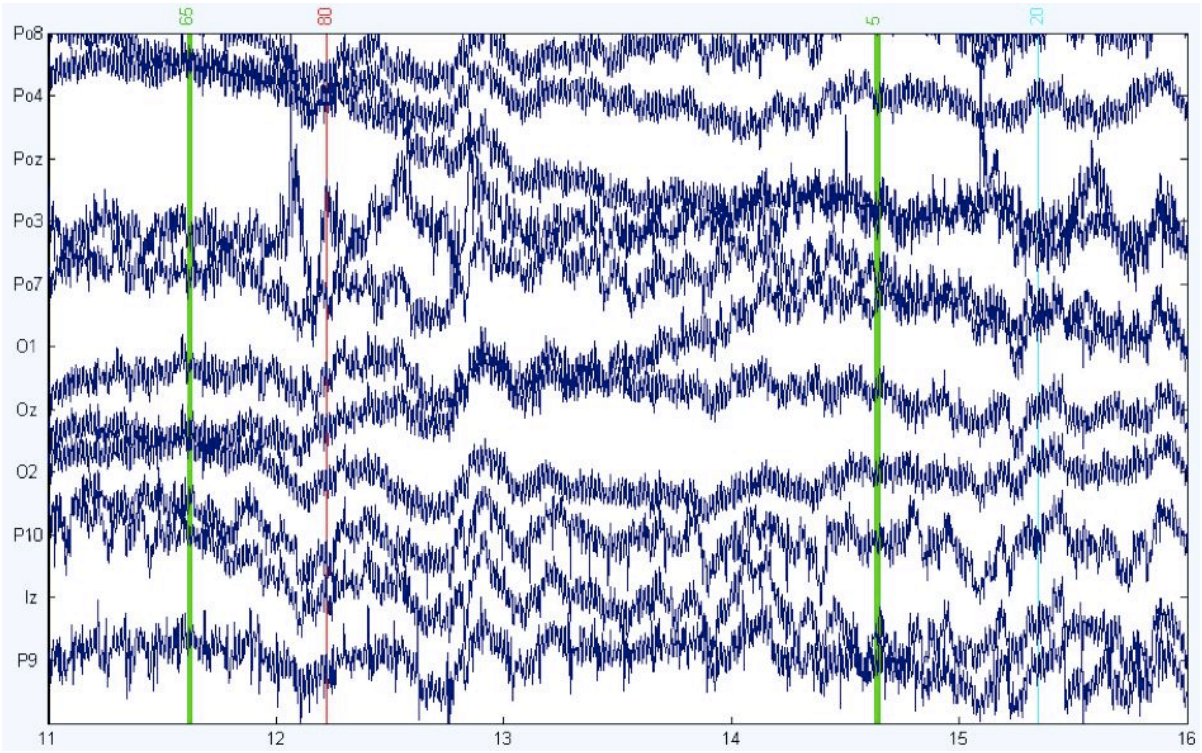


Figure 20. A sample of EEG that shows 10 electrodes over 5 seconds The numbers across the top of the graph are the events sent from the game.

EEGLab was then used to parse out the different event epochs into windows of data, the time before and after the stimulus is presented, and targets, the time before and after a subject reacts to the stimulus. Linear regression was then used to analyze these signal patterns and targets for the most important components in the signal.

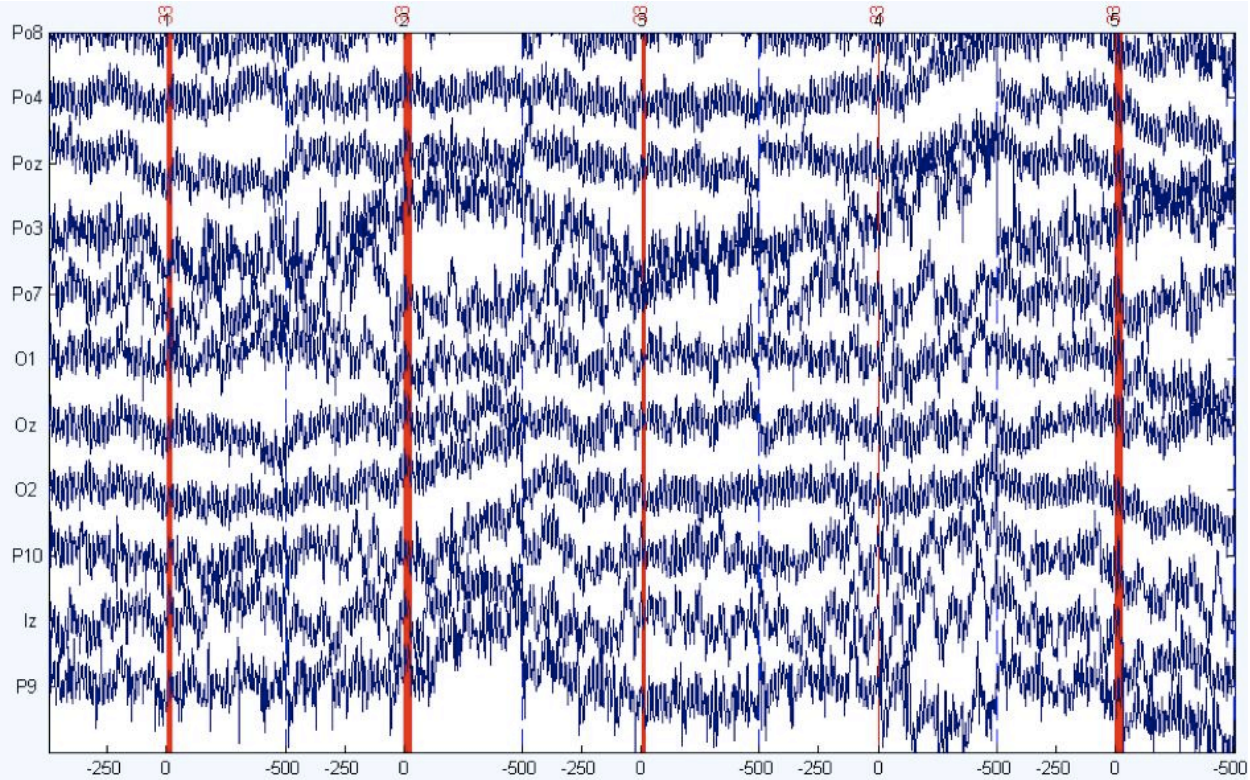


Figure 21. This graph depicts 500ms before and 500ms of the 5 separate trials of the same event

4.5 Feature Extraction

For this study the Matlab interactive stepwise linear regression gui was used to analyze the data and acquire inputs for the ANN. The interactive stepwise linear regression gui (SLRG) takes a column matrix and a row vector as inputs. In the column matrix ever column represents a trial and every row represents a dependent variable related to the trial. Every element in the row vector represents an independent variable and is related to the corresponding column in the column matrix. Using the data provided the SLRG walks the user through creating a model of

independent variables best representing the data. Every 'step' the user takes adds another independent variable to the model. The independent variables are chosen by their correlation to the dependent variables.

One result of the stepwise function is the p-value. The p-value is the probability of the test statistic being at least as extreme as the one observed given that the null hypothesis is true. Once the stepwise function was run the results with the highest p-value were chosen.

The advantage of linear regression is that reasonable estimates of the data can be calculated with very small sample sets. The disadvantage is that if the data is nonlinear it is difficult to find a model that fits the data well.

The test subjects were combined to find, for each electrode, what time samples had the highest correlation to the stimulus. The dependent variables are the EEG signals from the 64 electrodes taken from a time window around the stimulus. The independent variables depend on what stimulus is presented. When the subject was presented a stimulus to react to this was coded as one variable and when the subject was not presented a stimulus this was coded as the other variable. The electrode and time periods with highest correlations will be used by the Artificial Neural Networks to classify their respective independent variables.

4.6 Signal Classification

The Matlab Artificial Neural Network toolbox will be used to classify the features from the EEG signals into the subjects' intentions. A generic ANN will used to classify the physical movement as observed by articulating the Wii remote and the imaginary data when the user thinks of moving the remote. The input nodes, hidden nodes, output nodes and training epochs will all be set manually. For every training epoch each training pattern is fed forward to the ANN

and classified. Then the error is calculated and is back propagated for neuron weight correction. A Matlab script will need to be written to randomize the training and testing data and to create and run the ANN.

5 EXPERIMENT

5.1 Overview

The purpose of the experiment to attempt to classify brain signals in relation to controlling a video game controller. The experiment is designed to test the possibility of an ANN classifying both real and imaginary joystick movements by using the signals from an EEG. For the scope of this study the ANN will only classify movement and non-movement signals.

5.2 Signal Acquisition

Real 3D videogame play will be used to elicit the desired ERP component from the subject. Many BCI studies involve 5 or less subjects [17][12] and some studies have as low as two subjects[25]. Due to time constraints this study will attempt to find correlations across 10 subjects.

The subject was attached to an EEG and sat in front of a monitor with a video game controller as shown in figure 10. The EEG's sample rate was set to 512 Hz. During the first phase the subject actually used the controller and maneuvered it in accordance to the displayed stimuli on the screen. During the second phase the subject thought about maneuvering the controller in accordance to the stimuli displayed on the screen however the subject did not have a controller. The subjects were asked to complete at least 150 trials and, at most, 300. The large number of trials is to try to glean statistical validity out from study. Other studies have used as low as 100 samples or less [25][12].

After all of the data is collected the trials will be split into one second windows 400 ms

before the stimulus and 600ms after. The purpose of this is to monitor each subject's reaction to the stimulus.

5.3 Feature Extraction

After all of the data has been collected stepwise linear regression will be used to extract the features in the signal with the highest correlations to their respective events. All of the left, right, up, and down trials for every subject are grouped together as event one and all of the null trials (where no stimulus was presented) will be grouped as event 2. For every electrode stepwise regression is used to calculate the time periods within the one second interval with the highest correlations. After the correlations are figured for each electrode stepwise regression will be run again on the results to figure which have the highest correlation overall. The 10 electrodes and time period pairs with the highest correlations will be used as inputs to the ANN. Using more than 10 electrode and time period pairs did not help the classifier in preliminary trial runs of the ANN.

5.4 Signal Classification

A generic feed-forward back propagation ANN was used to classify the trials. The goal of the ANN was two-fold. The first goal was to detect if the subject manipulated the Wii remote in reaction to the stimulus or not. The second goal was to detect whether or not the subject imagined manipulating the Wii remote in reaction to the stimulus. The ANN had 10 input nodes, 25 hidden nodes, 25 training epochs, and 1 output node. The data was parsed so every trial only contained the sample for the 10 electrode and time period pairs. Then 80% (4599 trials) of the trials are used to train the ANN and 20% (731 trials) will be used to test it. To classify the

movement and imaginary trials the ANN was run 50 times. For each run the training and testing samples were chosen at random.

6 Experimental Results

6.1 Feature Extraction Results

The 10 electrode / time period combinations with the highest correlation were taken from both the movement and the imaginary trial data. The times presented are all in milliseconds after the stimulus was presented.

Table 1. Top 10 electrode and time period combinations from the movement data

Electrode	FCz	FC1	C2	CP4	P3	P1	P2	P4	LZ	L9
Sample (Hz)	482	361	463	346	419	350	459	435	465	432

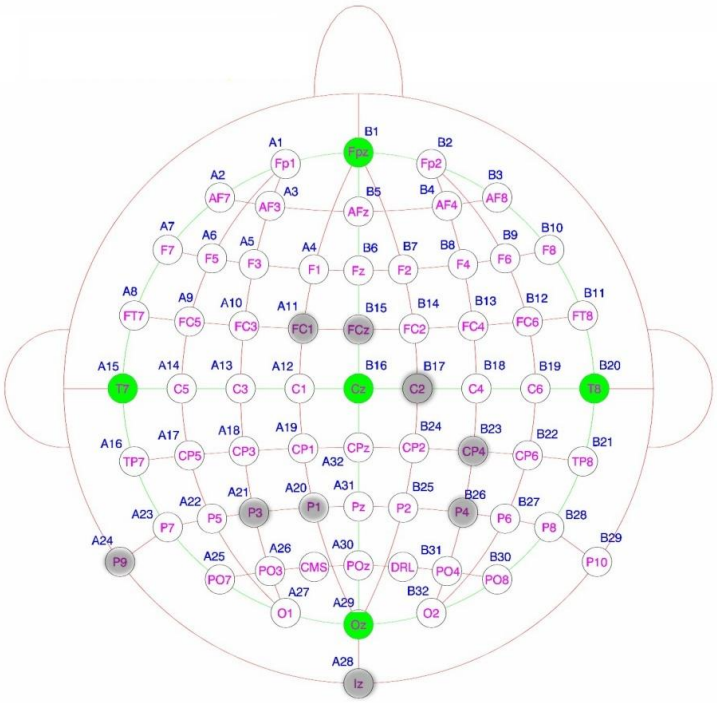


Figure 22. A head-map of the chosen electrodes, marked in gray, for the movement data

Table 2. Top 10 electrode and time period combinations from the imaginary data

Electrode	FCz	C1	C4	CPz	CP1	PO4	PO4	POz	P9	P9
Sample (Hz)	348	306	301	375	280	262	304	339	390	455

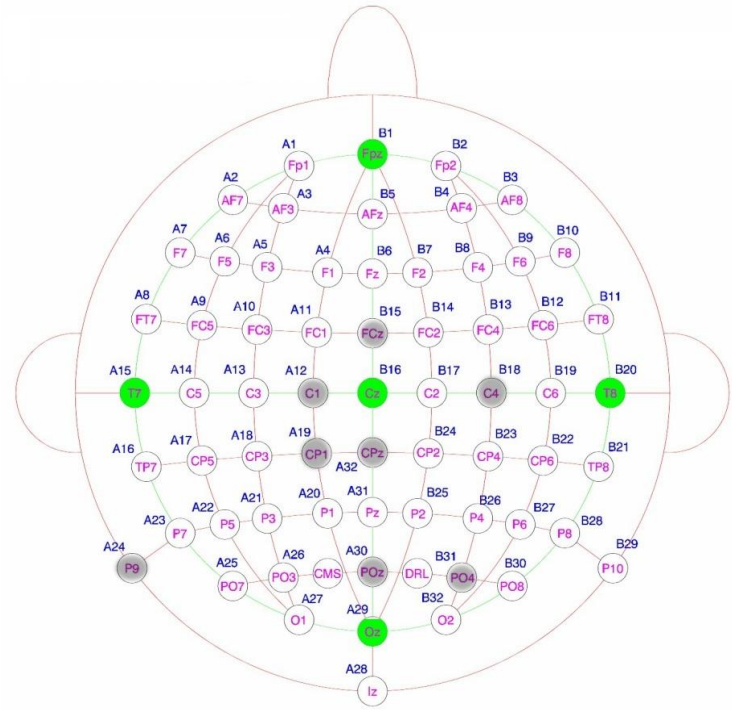


Figure 23. A head-map of the chosen electrode, marked in gray, for the imaginary data

6.2 Artificial Neural Network Results

The Neural Network test was based on how close it came to the right answer. The movement conditions were coded as 0 and the non-movement conditions were coded as 1. If the result was output < 0.5 it would be considered a movement and output > 0.5 considered a non-movement.

After running the ANN 50 times with the movement inputs the highest result was 75%, the lowest was 15% and the average result was 66%. With the imaginary data the highest result

was 65%, the lowest was 10%, and the average was 60%.

Forsland conducted a similar study in 2003 and the results can be compared to these. Forsland also used ANNs as the classification system but actually conducted the experiment differently. His study used 4 subjects and each was required to react to a stimulus. The subjects' EEG was monitored over 6 electrodes at 256 Hz. For 30 minutes each subject had to physically manipulate a joystick, left, right, up, and down according to the stimulus and for another 30 minutes they had to imagine manipulating a joystick. Even though every subject was analyzed individually the ANNs were tasked to learn to differentiate each movement. Overall the ANNs had a 77.5% success rate when classifying the physical movement data and 54.4% success rate when classifying the imaginary data.

Forsland had better results when classifying the physical movement data which might be because the Wii is more physically intense than a joystick. With the Wii remote the players had to grip the controller with both hands and move their forearms, wrists, and hands to perform the appropriate gesture, see figure 19. Forsland's joystick only had to be gripped by one hand and with an experienced player only a flick of the wrist can perform the proper gesture. The extra movement of the Wii remote could possibly be causing more noise in the EEG. This study has slightly better results when sampling the imaginary data. This could be due to the fact that this study used more electrodes, 10 compared to 6 or the higher sampling rate, 512 compared to 256. This study also used more subject 12 compared to 4 which could have made for a better sampling of data.

7 Conclusions

7.1 Artificial Neural Network Performance

The ANNs performed better than chance while only using 10 out of 32, 768 samples for both the physical movements and the imaginary data. This proves the hypothesis that a subject playing a 3D videogame will elicit brain states that can be classified by an Artificial Neural Network. I believe with further research the ANNs can be trained to have an even better success rate.

7.2 Blender and GlovePie performance

The use of Bender with its Python script interface and the GlovePie emulator served as a distinct advantage. GlovePie made creating scripts for the Wii Remote and sending the controller's status across the internet seamless. Blender's Python script interface made it easy to interpret the Wii Remote's status by monitoring the internet packets. The Blender's Python script interface also made sending signals to the EEG device very simple.

7.3 Electrode placement

A comparison of the chosen electrodes from both the physical movement and the imaginary data shows many similarities. Particularly, electrodes FCz and P9 are positively correlated with both the physical movement and the imaginary data. The areas around Cz and POz are also active in both data sets. This data will aid in electrode placement in future studies.

8 Future Works

8.1 Refine Feature Extraction

This study has proven the dataset does contain important features so future studies can be conducted to improve feature extraction. Alternate techniques like principal component analysis, autoregressive modeling, and Fast Fourier Transform can also be applied to the data for better feature extraction. With improved features the results of the classification will be improved.

8.2 Compare Classification Techniques

Artificial Neural Network is not the only classification systems used successfully with BCI. Linear discriminant analysis, Support Vector Machines, and Least Squared Regression have all been used recently used for BCI systems [1][24][28].

There are also many ANN configurations that need to be performed to find the optimal solution. Another experiment needs to be run to find the optimal topology for the ANNs. A comparison of different numbers of hidden layers and input nodes would lend insight on what ANN to use in an online (real-time classification) BCI study.

Different outputs should also be explored. Using an output of 0 and 1 can lead to data skewed toward the 0 output. By setting the output to be between -1 and 1 there is a chance the ANNs will classify the signal better.

8.3 Electrode Placement

The data gathered from the linear regression can be used to aid in electrode placement selection. Using fewer electrodes could help in three different ways. One, less electrodes means

less setup time for each subject. Two, using fewer electrodes could potentially make it computationally easier and faster for a classification system to process the data. Last, fewer electrodes could help eliminate the noise from using too many data points. If less electrodes are needed then this could help from processing extraneous data.

9 LITERATURE CITED

[1] Basar, Erol. Brain Oscillations. Principles and Approaches. Vol. 1. 2 vols. Berlin: Springer-Verlag, 1998.

[2] Bimber, Oliver. "Totall Recall " Computer 2008: 32 – 33.

[3] BIOSEMI. "Products". 2010. <http://www.biosemi.com/>.

[4] Birbaumer, Niels, et al. "The Thought Translation Device (Ttd) for Completely Paralyzed Patients " IEEE Transactions On Rehabilitation Engineering 8.2 (2000): 190 - 93.

[5] Bose, N. K., and P. Liang. Neural Network Fundamentals with Graphs, Algorithms, and Applications. Vol. New York: McGraw-Hill, 1996.

[6] Brain Master. "The International 10 - 20 System". 2010. <http://www.brainmaster.com/generalinfo/electrodeuse/eegbands/1020/1020.html>.

[7] Carmena, Jose M., et al. "Learning to Control a Brain-Machine Interface for Reaching and Grasping by Primates." Public Library of Science: Biology 1.2 (2003). Oct. 13 <http://biology.plosjournals.org/perlserv?request=get-document&doi=10.1371/journal.pbio.0000042>.

[8] Chapin, John K., et al. "Real-Time Control of a Robot Arm Using Simultaneous Recorded Neurons in the Motor Cortex." Nauture neuroscience 2.7 (1999): 664-70.

[9] Cichocki, Andrzej, et al. "Noninvasive Bcis: Multiway Signal-Processing Array Decompositions." Computer 2008: 34 -42.

[10] Dornhege, Guido. "Increasing Information Transfer Rates for Brain-Computer Interfacing." Universit at Potsdam, 2006.

[11] Farwell, L.D, and E. Donchin. "Talking Off the Top of Your Head: Toward a Mental Prosthesis Utilizing Event-Related Brain Potentials." Electroencephalography and clinical Neurophysiology 70.6 (1988): 510-23.

[12] Forslund, Pontus. "A Neural Network Based Brain-Computer Interface for Classification of Movement Related Eeg." Linköping University, 2003.

[13] Haykin, Simon. Neural Networks a Comprehensive Foundation. Upper Saddle River: Prentice Hall, 1999.

[14] Kennedy, P.R., and R.A Bakay. "Restoration of Neural Output from a Paralyzed Patient by a Direct Brain Connection." Neuroreport 9.8 (1998): 1707-12.

[15] Lalor, E. C., et al. "Steady-State Vep-Based Brain-Computer Interface Control in an

Immersive 3D Gaming Environment." EURASIP Journal on Applied Signal Processing 2005.19 (2005): 3156 - 64.

[16] Lecuyer, Anatole, et al. "Brain-Computer Interfaces, Virtual Reality, and Videogames." Computer 2008: 66 - 72.

[17] Leuthardt, Eric C, et al. "A Brain–Computer Interface Using Electrocorticographic Signals in Humans." Neural Engineering 1 (2004): 63-71.

[18] McFarland, Dennis, and Jonathan Wolpaw. "Brain-Computer Interface Operation of Robotic and Prosthetic Devices." 2008: 32 - 33.

[19] MettingVanRijn, A. C., A. Peper, and C. A. Grimbergen. "Amplifiers for Bioelectric Events: A Design with a Minimal Number of Parts." Medical & Biological Engineering & Computing.32 (1994): 305-10.

[20] Michael D. Rugg, and Michael G. H. Coles. Electrophysiology of Mind: Event-Related Brain Potentials and Cognition. Oxford Psychology Series Vol. NO. 25. Oxford Oxford University Press, 1995.

[21] Middendorf, M., et al. "Brain-Computer Interfaces Based on the Steady-State Visual-Evoked Response." IEEE Transactions On Rehabilitation engineering 8.2 (2000): 211 – 14.

[22] Nijholt, Anton, and Desney Tan. "Playing with Your Brain: Brain-Computer Interfaces and Games." ACM International Conference Proceeding Series 203 (2007): 305 - 06.

[23] Penny, William D., Stephen J. Roberts, and Maria J. Stokes. Imagined Hand Movements Identified from the Eeg Mu-Rhythm. London: Department of Electrical Engineering, 1998.

[24] Pfurtscheller, G., et al. "Current Trends in Graz Brain-Computer Interface (Bci) Research." Transactions on Rehabilitation Engineering 8.2 (2000): 216 - 19.

[25] Polak, Mark John. "Adaptive Logic Networks in a Brain-Computer Interface System." University of Alberta, 2000.

[26] Potts, A. M., and T. Nagaya. "Studies on the Visual Evoked Response. 1. The Use of the 0,06 Degree Red Target for Evaluation of Foveal Function." Invest Ophthalmol.4 (1965): 303 - 09.

[27]Olmsted, David D. "History and Principles of Neural Networks to 1960". 2006. <http://neurocomputing.org/NNHistoryTo1960.aspx>.

[28] Schalk, G., et al. "Bci2000: A General-Purpose Brain-Computer Interface (Bci) System." IEEE Transactions On Biomedical engineering 51.6 (2004): 1034 - 43.

[29] Shepherd, Gordon M. Foundations of the Neuron Doctrine (History of Neuroscience, No.

6). 1991.

[30] Stix, Gary. "Jacking into the Brain." Scientific American November 2008 2008: 56 - 61.

[31] U.S. National Institutes of Health. "Inroduction to the Nervous System". 2010. SEER Training Modules. US. National Institutes of Health2010. <http://training.seer.cancer.gov/anatomy/nervous/>.

[32] Utts, Jessica, and Robert Heckard. Mind on Statistics Ed. Carolyn Crockeet. Vol. 1. Pacific Grove: Duxbury, 2002.

[33] Wolpaw, Jonathan, et al. "The Wadsworth Center Brain-Computer Interface (Bci) Research and Development Program." IEEE Transactions on Neural Systems and Rehabilitation Engineering 11.2 (2003): 204 – 07.

[34] Wolpaw, J.R., D.J. McFarland, and T.M. Vaughan. "Brain-Computer Interface Research at the Wadsworth Center." IEEE Transactions On Rehabilitation Engineering 8.2 (2000): 222 - 26.

10 APPENDIX

A. Matlab Code

```
%a function to parse a persons EEG data by the different movements
function  eeg_manip_moves(subjectName, path, numberOfDatasets, range, baselinePath,
baselineSave)

x = [-0.5 0.5];
leftset = [];
savePathLeft1 =[];
savePathRight1 =[];
savePathUp1 =[];
savePathDown1 =[];

for i = 1:numberOfDatasets
   %creating a string for the file name
file = strcat(char(subjectName),'_',int2str(i),'of',int2str(numberOfDatasets),'_64.set');

   %loading the dataset
preRef = pop_loadset(char(file),char(path));
   %rerefrencing to an average reference
postRef = pop_reref( preRef, [], 'refstate',0);

        %run the data through a high and low pass filter
postBandFilter = pop_eegfilt( postRef, 1, 30, [], [0]);
filt_Nof4 = pop_eegfilt( postBandFilter, 0, 30, [], [0]);
```

```
%using EEGLAB to parse the raw EEG into the respetive movements

left = pop_epoch(filt_Nof4,{'5'}, range, 'newname', 'Continuous EEG Data ', 'epochinfo', 'yes');

right = pop_epoch(filt_Nof4,{'3'}, range, 'newname', 'Continuous EEG Data ', 'epochinfo', 'yes');

up = pop_epoch(filt_Nof4,{'65'}, range, 'newname', 'Continuous EEG Data ', 'epochinfo', 'yes');

down= pop_epoch(filt_Nof4,{'129'}, range, 'newname', 'Continuous EEG Data ', 'epochinfo',
'yes');

null = pop_epoch (filt_Nof4,{'33'},range,'newname','Continuous EEG Data','epochinfo','yes');


savenameLeft =

strcat([char(subjectName),'_',int2str(i),'of',int2str(numberOfDatasets),'_64_left']);

savenameRight =

strcat([char(subjectName),'_',int2str(i),'of',int2str(numberOfDatasets),'_64_right']);

savenameUp = strcat([char(subjectName),'_',int2str(i),'of',int2str(numberOfDatasets),'_64_up']);

savenameDown =

strcat([char(subjectName),'_',int2str(i),'of',int2str(numberOfDatasets),'_64_down']);

savenameNull =

strcat([char(subjectName),'_',int2str(i),'of',int2str(numberOfDatasets),'_64_null']);


%the pop_export command converts the file to ASCII

pathL = strcat([char(path), '\left_move\']);

mkdir(pathL)

pop_export(left,strcat(pathL,savenameLeft),'ica','off','time','off','elec','off');%,['off'],['off'],1E-
3,['off'])
```

```
pathR = strcat([char(path),'\right_move\']);
mkdir(pathR);
pop_export(right,strcat(pathR,savenameRight),'ica','off','time','off','elec','off');

pathU = strcat([char(path),'\up_move\']);
mkdir(pathU);
pop_export(up,strcat(pathU,savenameUp),'ica','off','time','off','elec','off');

pathD = strcat([char(path),'\down_move\']);
mkdir(pathD);
pop_export(down,strcat(pathD,savenameDown),'ica','off','time','off','elec','off');

pathNull = strcat([char(path),'\null_move\']);
mkdir(pathNull);
pop_export(null,strcat(pathNull,savenameNull),'ica','off','time','off','elec','off');


if(i == 1)

    savePathLeft1 = strcat(pathL,savenameLeft);
    savePathRight1 = strcat(pathR,savenameRight);
    savePathUp1 = strcat(pathU,savenameUp);
    savePathDown1 = strcat(pathD,savenameDown);
    savePathNull1 = strcat(pathNull,savenameNull);


elseif(i == numberOfDatasets)
```

```
%load both saved (ASCII) datasets,concatenate and save them
left1 = load(savePathLeft1);
left2 = load(strcat(pathL,savenameLeft));
totalLeft = horzcat(left1,left2);

allTrials(totalLeft,'allLeftTrials',pathL);
save(strcat(pathL,'totalLeft.dat'),'totalLeft');

right1 = load(savePathRight1);
right2 = load(strcat(pathR,savenameRight));
totalRight = horzcat(right1,right2);

allTrials(totalRight,'allRightTrials',pathR);
save(strcat(pathR,'totalRight.dat'),'totalRight');

 up1 = load(savePathUp1);
 up2 = load(strcat(pathU,savenameUp));
 totalUp = horzcat(up1,up2);
allTrials(totalUp,'allUpTrials',pathU);
save(strcat(pathU,'totalUp.dat'),'totalUp');

 down1 = load(savePathDown1);
 down2 = load(strcat(pathD,savenameDown));
 totalDown = horzcat(down1,down2);
allTrials(totalDown,'allDownTrials',pathD);
save(strcat(pathD,'totalDown.dat'),'totalDown');
```

```
    null1 = load(savePathNull1);
    null2 = load(strcat(pathNull,savenameNull));
    totalNull = horzcat(null1,null2);
   allTrials(totalNull,'allNullTrials',pathNull);
   save(strcat(pathNull,'totalNull.dat'),'totalNull');

end
end

                    %a function to save out eeg samples as a line of ASCII
function allTrials(set,fileName,path)

%data was sampled at 512hz and each sample should be exactly 1 second
%each trial should be 512 points
totalTrials = length(set)/512;

pathAllElec = strcat(path,'allEachElectrode\');
mkdir(pathAllElec);

for j = 1:64
   totalSet(1:512) = 0;
   currentSet = set(j,1:length(set));

   for i=0:totalTrials -1
```

```
        %length(x1((512 * i)+1:(512 * i)+512))
        v1 =(512 * i)+1;
        v2 =(512 * i)+512;
        if(i ==1)
            totalSet =  currentSet(v1:v2);

        else
            totalSet = [totalSet;  currentSet(v1:v2)];
        end
    end

    currentRecord = int2str(j);

    filename = strcat(fileName,currentRecord,'.dat');
    dlmwrite(strcat(pathAllElec,filename),totalSet);

end

%A function to run Matlab's stepwise regression suite
%independent move and dependent move are the respective sets for the Left, right, up, and down
movements. The non moves were when the subject was asked to hold the controller still
function  runStepwise (independentMove,independentNonMove, dependentMove,
dependentNonMove)
```

```
totalIndependent = [independentMove';independentNonMove'];

totalDependent = [dependentMove;dependentNonMove];

stepwise(totalIndependent,totalDependent);

end


        % a function to run the ANN on the whole group's data
function  runNN2 (xVars,yVars)


   totalTrialAmt = length(xVars);

   trainTrialAmt = uint16(totalTrialAmt * .8);

   testTrialAmt = uint16(totalTrialAmt *.2);


   movementTrainAmt = trainTrialAmt / 2;

   movementTestAmt  = testTrialAmt / 2;

   %1426 were the total amount of trials
   trainPatsA = (1426 -movementTestAmt);

   trainPatsB = (totalTrialAmt-movementTestAmt);


   trainTarsA = (1426 -movementTestAmt);

   trainTarsB = (totalTrialAmt-movementTestAmt);


   trainPats = [xVars(1:trainPatsA,:);xVars(1426:trainPatsB,:) ];

   trainTars = [yVars(1:trainTarsA,:);yVars(1426:trainTarsB,:) ];
```

```
a = trainPatsA;

b = trainPatsB;


testPats = [xVars(trainPatsA:1426,:);xVars(trainPatsB+1:totalTrialAmt,:)];

testTars = [yVars(trainTarsA:1426,:);yVars(trainTarsB+1:totalTrialAmt,:)];


% initializing the ANN
net = newff(minmax(trainPats'),[25 1]);


net.trainParam.epochs = 20;


[net,tr] = train(net,trainPats',trainTars');



a = sim(net,testPats');

diff = a - testTars';

right = 0;

wrong = 0;

for i = 1:length(diff)

   if(abs(diff(i)) < .5)

      right = right + 1;

   else

      wrong = wrong + 1;

   end

end

length(testTars)
```

```
    stats = [right;wrong]
end

%
```

B. PieGlove Code

```
// Set keys to wiimote/nunchuk buttons

var.up = 0

var.down = 0

var.left = 0

var.right = 0

var.a = 0

var.b = 0

var.c = 0

var.one = 0

var.two = 0

var.home = 0

var.minus = 0

var.plus = 0

var.pitchRight = 0

var.pitchLeft = 0

var.rollForward = 0

var.rollBack = 0
```

```
if Wiimote.B or Wiimote.Classic.b then
  var.b = 1
endif

if RemoveUnits(Wiimote.Pitch) > 45
  var.pitchRight = 1
endIf

if RemoveUnits(Wiimote.Pitch) < -45
  var.pitchLeft = 1
endIf
if RemoveUnits(Wiimote.Roll) > -40 && RemoveUnits(Wiimote.Roll) < 30
  var.rollForward = 1
endIf
if RemoveUnits(Wiimote.Roll) < -130 || RemoveUnits(Wiimote.Roll) > 150
  var.rollBack = 1
endIf

// Set the middle two LEDs to ON
if Wiimote.Exists then
  Wiimote.Led1 = false
  Wiimote.Led2 = true
  Wiimote.Led3 = true
  Wiimote.Led4 = false
endif
```

```
// Show expansion and wiimote forces
//debug = 'Bat='+Wiimote.Battery+', Pitch='+RemoveUnits(Wiimote.SmoothPitch)+',
Roll='+RemoveUnits(Wiimote.SmoothRoll)+', '+Wiimote.RelAccX+', '+Wiimote.RelAccY+',
'+Wiimote.RelAccZ+'  '+var.up+' '+var.down+' '+var.left+' '+var.right+' '+var.one+' '+var.two+'
'+var.home+' '+var.plus+' '+var.minus+' '+var.a+' '+var.b+' '+var.c

//debug = 'Bat='+Wiimote.Battery+', Pitch='+RemoveUnits(Wiimote.Pitch)+',
Roll='+RemoveUnits(Wiimote.Roll)+', '+Wiimote.RelAccX+', '+Wiimote.RelAccY+',
'+Wiimote.RelAccZ
debug =   'Pitch Right =' + var.pitchRight + ', Pitch Left =' + var.pitchLeft+', Roll Forward ='+
var.rollForward+ ', Roll Back'+ var.RollBack

//SendOsc("localhost", 4950, "/", RemoveUnits(Wiimote.SmoothPitch),
RemoveUnits(Wiimote.SmoothRoll), Wiimote.RelAccX, Wiimote.RelAccY, Wiimote.RelAccZ,
var.up, var.down, var.left, var.right, var.one, var.two, var.home, var.plus, var.minus, var.a, var.b,
var.c, Wiimote.Battery)
SendOsc("localhost", 4950, "/", var.pitchRight, var.pitchLeft,var.rollForward,var.rollBack)
//debug = Wiimote.Led1
//wait 1000 ms
```

C. Blender Code

```
% This script controls the stimuli
import GameLogic
```

```
import random


#!/usr/bin/python2.5



cont = GameLogic.getCurrentController()

own = cont.getOwner()

#my_ball_own = cont.getActuator("end_ball").getOwner()



#Actuator

m_Left = cont.getActuator("m_Left")

m_Right = cont.getActuator("m_Right")

m_Up = cont.getActuator("m_Up")

m_Down = cont.getActuator("m_Down")

presented = cont.getActuator("presented")



if own.setPins == 0:

        print 'stimulus_brain'

        for x in range(0, 8):

                print "Getting bit %d: %d" % (x, getBit(lpt, x))
```

```
        own.setPins = 1
        for x in range(0, 8):
                setBit(lpt,x,0)

#stimulus logic
if own.timer > 1 and own.shoot == 1:


        direction = random.randint(1,5)
        if direction == 1:
                shoot = cont.getActuator("left_shoot")
                GameLogic.addActiveActuator(shoot,1)
                own.shoot = 0
                own.setPins = 0
                setBit(lpt,0,1)
                setBit(lpt,6,1)
                GameLogic.addActiveActuator(m_Left,1)
                GameLogic.addActiveActuator(presented,1)
                print '****left shoot****'


        if direction == 2:
                shoot = cont.getActuator("right_shoot")
                GameLogic.addActiveActuator(shoot,1)
                own.shoot = 0
                own.setPins = 0
```

```
        setBit(lpt,1,1)
        setBit(lpt,6,1)
        GameLogic.addActiveActuator(m_Right,1)
        GameLogic.addActiveActuator(presented,1)
        print '****right shoot****'

if direction == 3:
        shoot = cont.getActuator("top_shoot")
        GameLogic.addActiveActuator(shoot,1)
        own.shoot = 0
        own.setPins = 0
        setBit(lpt,2,1)
        setBit(lpt,6,1)
        GameLogic.addActiveActuator(m_Up,1)
        GameLogic.addActiveActuator(presented,1)
        print '****top shoot****'

if direction == 4:
        shoot = cont.getActuator("bottom_shoot")
        GameLogic.addActiveActuator(shoot,1)
        own.shoot = 0
        own.setPins = 0
        setBit(lpt,3,1)
```

```
            setBit(lpt,6,1)
            GameLogic.addActiveActuator(m_Down,1)
            GameLogic.addActiveActuator(presented,1)
            print '****bottom shoot****'

    if direction == 5:
            own.shoot = 0
            own.setPins = 0
            setBit(lpt,4,1)
            setBit(lpt,6,1)
            #GameLogic.addActiveActuator(m_Down,1)
            GameLogic.addActiveActuator(presented,1)
            print '****Nothing shoot****'

if own.timer > 3:
    own.shoot = 1
    own.timer = 0
```

%%%%%%

This script will receive the wii remote data from the current Ethernet packet, interpret the packet Make the appropriate move on screen, and send the data to the EEG to mark the data

```
import GameLogic
import random
```

```
import Rasterizer
import math
import socket
import OSC

cont =GameLogic.getCurrentController()
own = cont.getOwner()

# sensors:
left = cont.getSensor("left")
right = cont.getSensor("right")
up = cont.getSensor("up")
down = cont.getSensor("down")
#Actuators:
motion = cont.getActuator("motion")
trackto = cont.getActuator("trackto")
move_Left = cont.getActuator("move_Left")
move_Right = cont.getActuator("move_Right")
move_Up = cont.getActuator("move_Up")
move_Down = cont.getActuator("move_Down")
moved = cont.getActuator("moved")

#setting all of the bits in the parallel port to 0
if own.setPins == 0:
        print 'ship_brain'
```

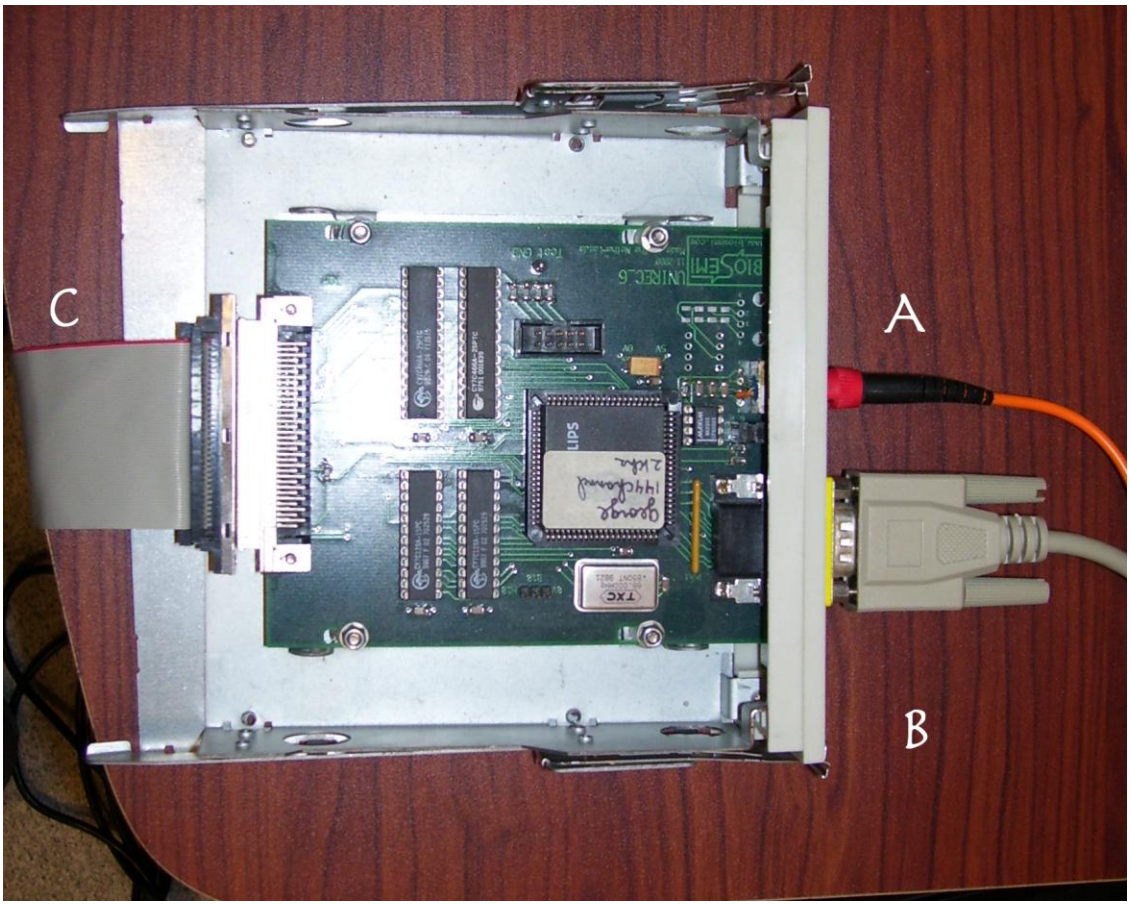
```
        for x in range(0, 8):
                print "Getting bit %d: %d" % (x, getBit(lpt, x))
        own.setPins = 1
        for x in range(0, 8):
                setBit(lpt,x,0)
#obtaining the OCS packet from the Localhost
def wii_data(host, port):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    #s.settimeout(0.01)
    s.bind((host, port))
    raw, addr = s.recvfrom(1024)
    dec = OSC.decodeOSC(raw)
    return dec

#parsing the packet into its individual packets
pitchRight,pitchLeft,rollForward,rollBack = 0,0,0,0
if own.one_move == 1 and own.wiiTimer >.1:
        pitchRight,pitchLeft,rollForward,rollBack = wii_data('localhost', 4950)
        own.wiiTimer = 0

#movement logic
motion.setDLoc(0,0.3,0,1)
if pitchLeft  and (own.one_move == 1) and (own.presented == 1):
        bound = GameLogic.getCurrentScene().getObjectList()["OBleft_bound"]
```

```
        trackto.setObject(bound)
        own.one_move = 0
        own.setPins = 0
        setBit(lpt,0,1)
        setBit(lpt,5,1)
        GameLogic.addActiveActuator(move_Left,1)
        GameLogic.addActiveActuator(moved,1)
        print '****left move****'

elif  pitchRight and (own.one_move == 1) and (own.presented == 1):
        bound = GameLogic.getCurrentScene().getObjectList()["OBright_bound"]
        trackto.setObject(bound)
        own.one_move = 0
        own.setPins = 0
        setBit(lpt,1,1)
        setBit(lpt,5,1)
        GameLogic.addActiveActuator(move_Right,1)
        GameLogic.addActiveActuator(moved,1)
        print '****right move****'

elif rollBack and (own.one_move == 1) and (own.presented == 1):
        bound = GameLogic.getCurrentScene().getObjectList()["OBtop_bound"]
        trackto.setObject(bound)
        own.one_move = 0
```

```
        own.setPins = 0
        setBit(lpt,2,1)
        setBit(lpt,5,1)
        GameLogic.addActiveActuator(move_Up,1)
        GameLogic.addActiveActuator(moved,1)
        print '****up move****'

elif rollForward and (own.one_move == 1) and (own.presented == 1):
        bound = GameLogic.getCurrentScene().getObjectList()["OBbottom_bound"]
        trackto.setObject(bound)
        own.one_move = 0
        own.setPins = 0
        setBit(lpt,3,1)
        setBit(lpt,5,1)
        GameLogic.addActiveActuator(move_Down,1)
        GameLogic.addActiveActuator(moved,1)
        print '****down move****'

elif own.one_move == 1:
        bound = GameLogic.getCurrentScene().getObjectList()["OBoriginal_bound"]
        trackto.setObject(bound)
if own.timer > 3:
        bound = GameLogic.getCurrentScene().getObjectList()["OBstart_bound"]
        startLoc = bound.getPosition()
```

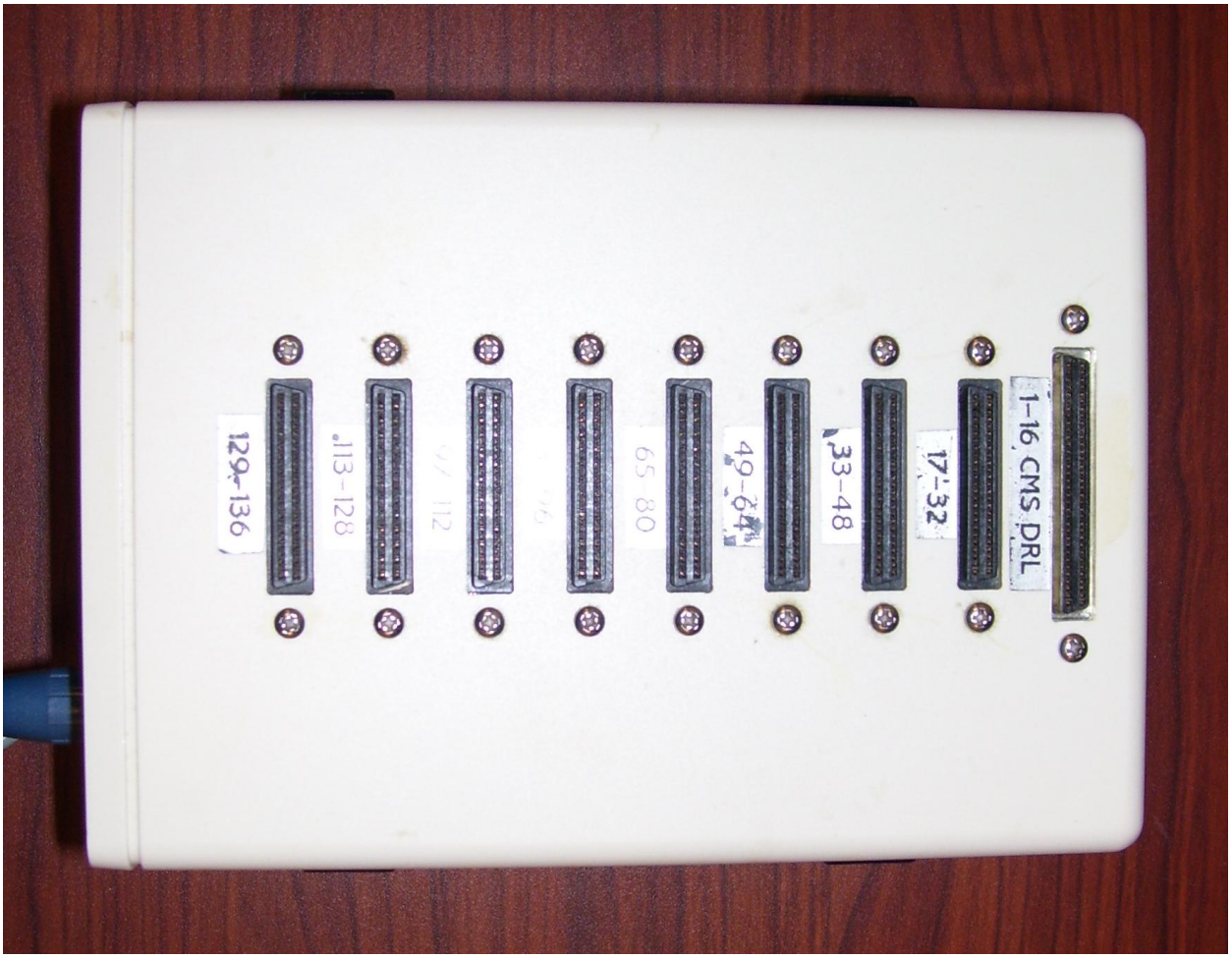
```
        own.setPosition(startLoc)
        own.timer = 0
        bound = GameLogic.getCurrentScene().getObjectList()["OBoriginal_bound"]
        trackto.setObject(bound)
        own.one_move = 1
        own.shoot =1
        own.presented = 0

GameLogic.addActiveActuator(trackto,1)
GameLogic.addActiveActuator(motion,1)
```

D. EEG equipment



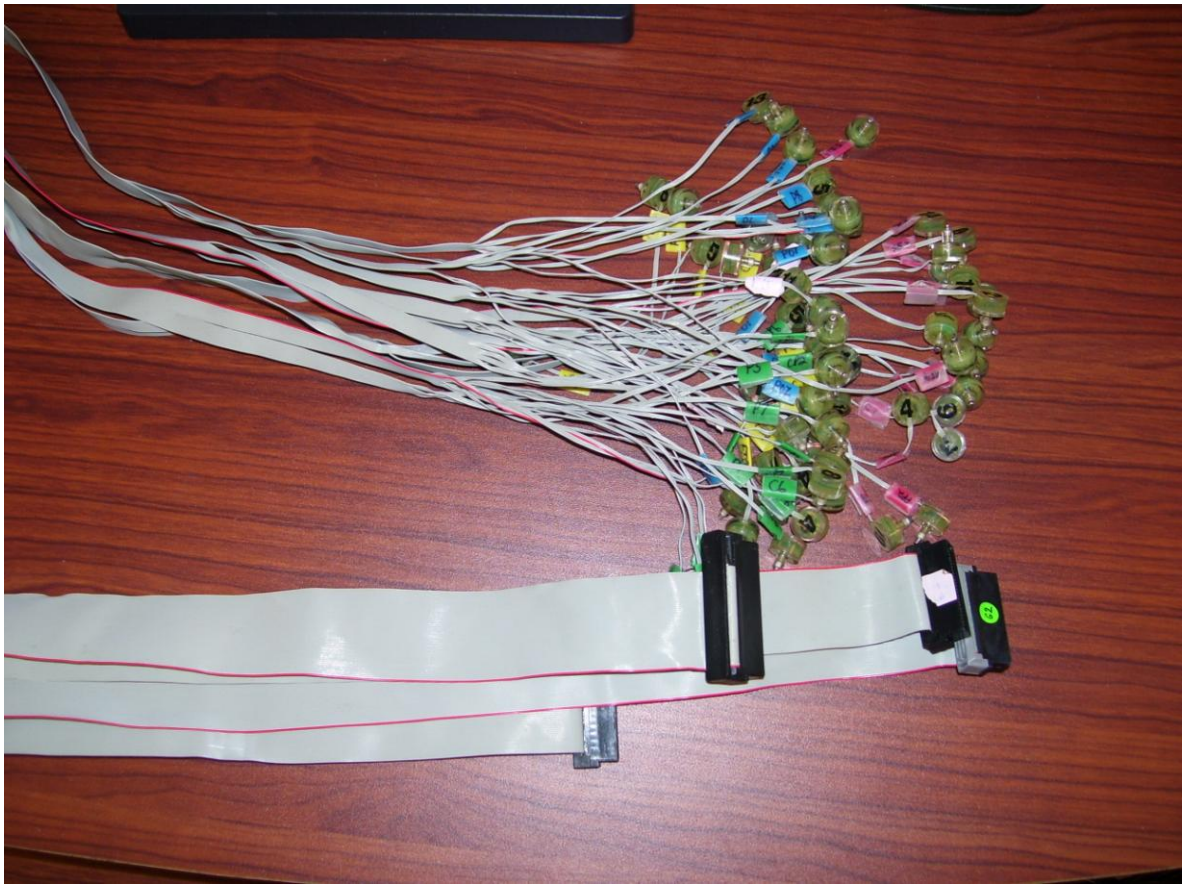
24. EEG Interface Device.

Figure 24 depicts the EEG Interface Device. A, is a fiber-optic cable that connects directly to the EEG Amplifier. B, is a serial cable that allows an application to send codes that will be marked in the EEG data in real time. C, shows the IDE cable that transmit the EEG amplifier data and the serial data to a computer to be recorded via the LabView software.



25. EEG Amplifier

26. A front view of the EEG amplifier. On top is the Amplifier and the battery is on bottom. To the left is charging station for the battery.



27. The individual electrodes that plug into the EEG amplifier.

28. Signa gel is placed between the electrodes and the scalp. The gel helps the electrodes better conduct electricity.