1-1-2011

Hybrid Sensing and Adaptive Control for Direct Brain Actuation of Artificial Limbs

Christopher Aasted
University of Denver

HYBRID SENSING AND ADAPTIVE CONTROL FOR DIRECT BRAIN ACTUATION OF ARTIFICIAL LIMBS

_____

A Dissertation

Presented to

the Faculty of Engineering and Computer Science

University of Denver

_____

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

_____

by

Christopher M. Aasted

June 2011

Advisor: Dr. Rahmat A. Shoureshi

Author: Christopher M. Aasted
Title: HYBRID SENSING AND ADAPTIVE CONTROL FOR DIRECT BRAIN ACTUATION OF ARTIFICIAL LIMBS
Advisor: Dr. Rahmat A. Shoureshi
Degree Date: June 2011

Abstract

Developing a non-invasive direct brain control of artificial limbs is both challenging and desirable. Such a sensory and control system, if successful, will have a profound impact on the disabled. In this dissertation, we present the design and development of a non-invasive, hybrid sensory system, which uses near-infrared spectroscopy (NIRS) and electroencephalography (EEG) to measure brain activity with simultaneous electromyography (EMG) to provide feedback data in a healthy limb. Through the combination of these sensory techniques, we have successfully trained a control system capable of mapping brain activity onto muscle actuation. The design of a control algorithm capable of automatic reconfiguration to account for changing sensor conditions, selection of an appropriate pre-trained network based on input characteristics, and adaptation to adjust output based on the user's activity are investigated. The selection of an appropriate algorithm and its initial performance using our sensory system are presented and discussed.

The sensory and control system are designed for application in artificial limb control for persons who have undergone amputation of an upper-extremity. Actuation of the elbow and wrist are the primary focus of the study, with the intent to expand to forearm torsion and hand grasping in subsequent studies. During the course of the investigation, the additional function of treating phantom limb pain was incorporated into the design, which has also lead to increased sensor resolution requirements.

Acknowledgements

I would like to thank my advisor, Dr. Rahmat Shoureshi, for all of the opportunities and guidance he has offered over the years. I would also like to thank my doctoral committee: Dr. Brad Davidson, Dr. David Gao, Dr. Siavash Pourkamali, Dr. Ramki Thurimella, and Dr. Yun-Bo Yi, for their time and assistance.

The research team would like to acknowledge the Defense Advanced Research Projects Agency, the Sensor System program of CMMI Division of the National Science Foundation, and the Bioscience Initiative at the State of Colorado, for the funding that made this research possible.

Table of Contents

List of Tables

List of Figures

CHAPTER ONE: INTRODUCTION

Problem Statement

This research aims to develop a non-invasive brain monitoring system capable of performing real-time control of artificial limbs. The primary application of this device will be for control of prosthetic limbs used by persons who have undergone partial or complete upper-extremity amputation. Any control algorithm used to accomplish these goals must be capable of operating in real-time, while minimizing the computational demands in order to keep hardware requirements in a cost effective range.

Background and Significance

As part of the goal of developing engineered systems that would improve the quality of life, this research is focused on the feasibility analysis, design, and prototyping of a non-invasive sensory system that detects human brain intentions about the movement of the musculoskeletal system and utilizes those signals to command and control artificial limbs, robotic systems, or powered exoskeletons that assist individuals with disabilities. It is estimated that 1.7 million people in the United States have experienced limb loss (Ziegler-Graham 2008, 422-429). In the CDC publication "Arthritis – At A Glance 2009" the current number of people in the United States with diagnosed arthritis is an estimated 46 million. This is further projected to reach 67 million by 2030 (Center for Disease Control and Prevention, 2009). The need for assistive devices to overcome chronic

disabilities is a growing market with substantial limitations in the current technology due to the complexity of fine motor control. Advancements in this field of study will have a substantial impact on the quality of life for millions of people. Many injuries, whether due to traffic accidents or other unfortunate events, cause children to lose hands and legs and be disabled for the rest of their lives. With improvements in medical technology, fewer major injuries result in fatality and as a result a greater percentage of people are living with limb loss. Amputations resulting from military action are of particular interest since the disabled persons would otherwise be young, healthy individuals. While these statistics are of major concern, there is yet to be a commercially available system that would provide natural mobility for these individuals, and current research in this field is generally focused on invasive methods of restoring limb function (Defense Sciences Office 2010).

Through the combination of electroencephalography (EEG), near-infrared spectroscopy (NIRS), and electromyography (EMG), a non-invasive control system utilizing an adaptation algorithm and feedforward/feedback sensor integration for real-time control of artificial limbs is being investigated. Through the integration of the temporal response of EEG with the spatial accuracy of NIRS, an accurate, non-invasive control system can be developed. Based on the EMG signals from healthy subjects, we are able to train the control system to properly correlate EEG and NIRS input patterns to matching muscle activations. A diagram of this combined feedback and adaptation control system is depicted in Figure 1.1. NIRS is used as a second feedback loop to continuously adapt the algorithm and reduce error. This research will hopefully enable people with disabilities to directly control artificial limbs and robotic arms to assist them

in their daily activities. It is without a doubt that many other application areas, such as forensic lie detection, can benefit from the outcomes of this research through advances in portable, inexpensive brain monitoring equipment.

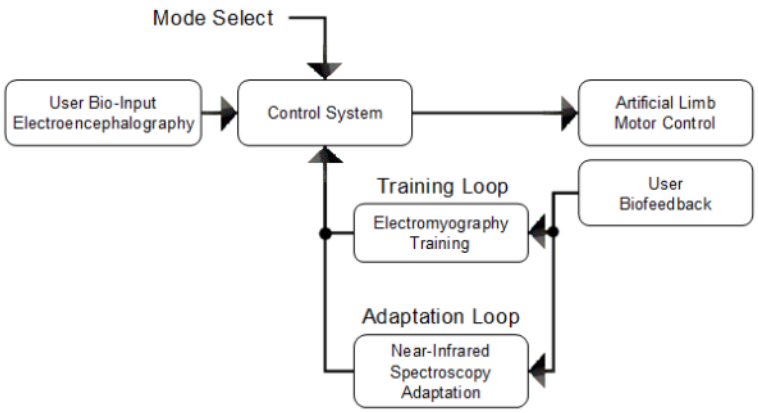


Figure 1.1: Training and adaptive loops for control of artificial limbs

Shortcomings of Current Technology

Most current prostheses are actuated using mechanical sensors and/or biosensors. Biosensors detect signals from the user's nervous or muscular systems, which are relayed to a controller located inside the device. Limbic and actuator feedback may be used as inputs to the function of the controller. Examples include wires that detect electrical activity on the skin, needle electrodes implanted in muscle, or solid-state electrode arrays with nerves growing through them. Mechanical sensors process aspects affecting the device including limb position, applied force, and load, and then relay this information to the biosensor or controller. A prosthesis controller may be connected to the user's nervous and muscular systems as well as the prosthesis itself. The controller may send intention commands from the user to the actuators of the device, and may interpret feedback from the mechanical and biosensors to the user. Myoelectric control of simple

joints is currently the most common advanced solution for prosthetic limbs. Many amputees still use systems that have no motor control or have simple attachments for hand prostheses.

In 2005, Defense Advanced Research Projects Agency (DARPA) started the Revolutionizing Prosthetics Program that had the goal of developing a neural-controlled prosthetic limb that will provide full motor and sensory capability to upper extremity amputee patients. The goals for the prosthesis are to be controlled, feel, look, and perform like the original limb, with a time line for project completion of less than four years. The limb would have to be wired directly into the peripheral nervous system in order to have enough dexterity to pick up a raisin or to write in longhand. Other system requirements include being sensitive enough for the wearer to handle day-to-day tasks in the dark, while being strong enough to lift 60 pounds at a time (Defensetech.org 2005). To accomplish these ambitious goals, DARPA started running the program in parallel with multiple companies across several related fields, all working on different aspects of the project at the same time. Two of the groups that received substantial funding and have published some of their results are the Applied Physics Laboratory at Johns Hopkins University and DEKA Research and Development Corporation.

The Johns Hopkins University Applied Physics Laboratory (APL) has developed a prototype of the first fully integrated prosthetic arm that can be controlled directly from the nervous system, provide sensory feedback, and utilize eight degrees of freedom. This is a substantial improvement over currently available systems. The APL prototype is called Proto 1 and its virtual environment system underwent clinical evaluations

conducted by team partners at the Rehabilitation Institute of Chicago (RIC) in January and February of 2008 (Kram 2008).

The advanced degree of natural control and integrated sensory feedback demonstrated with Proto 1 are enabled by Targeted Muscle Reinnervation (TMR), a technique developed at RIC that uses the transfer of residual nerves from an amputated limb to unused muscle regions in areas near the injury. In the case of one human subject, Jesse Sullivan, the nerves were transferred to the pectoral area of his chest (Kram 2008).

During clinical evaluation of the limb at RIC, Jesse Sullivan displayed substantial improvements in testing, including the ability to reposition his thumb for different hand grasping positions, remove a credit card from a pocket, stack cups while adjusting the strength of the grip via sensory feedback, and to walk using the free-swing mode of the limb (Kram 2008).

APL and other team members began working on a second prototype in 2008, as part of a continuing Revolutionizing Prosthetics program. The new design will have more than 25 degrees of freedom as well as the strength and speed to approach the capabilities of a human limb. These new performance specifications will be combined with more than 80 individual sensory elements for feedback of touch, temperature, and limb position (Kram 2008).

Another development is the functional demonstration of Injectable MyoElectric Sensor (IMES) devices, which are very small injectable or surgically implantable devices used to measure muscle activity at the source, as opposed to surface electrodes on the skin that were used during testing of the first prototype (Kram 2008).

DARPA is simultaneously funding the development of a competing arm at DEKA. The DEKA arm meets the weight requirement at less than 8 pounds, while making use of 12 microprocessors that monitor pressure sensors in the hand and actuate rumble motors at the joining with the patient's body. Their arm does not currently make use of a direct nerve interface (Virtualworldlets.net 2008).

While the research currently being conducted in this field certainly has the potential to outperform existing prosthetic limb systems, the focus of the research is on highly sophisticated replacement limbs that require invasive means of providing control. This project aims to develop a system that incurs less risk for the user and is a more affordable solution.

Planned Research

In order to accomplish the goals of this research, the following tasks will be completed (Figure 1.2): Following initial research into available sensor technologies, appropriate equipment will be purchased to conduct preliminary studies using healthy subject data. A simple control system will be completed in order to validate that the imaging techniques used have the potential to meet the requirements of the study. Based on the shortcomings of the equipment used in the preliminary studies, a new hybrid sensor system will be constructed to test the ability of the combined imaging techniques. From data collected using the hybrid system, an investigation will be carried out to select an appropriate control algorithm for this application and a suitable control system will be completed. Based on the performance of the prototype sensory system, improvements will be made for the design of a sensor device sufficient for final application.

Pending positive results from this study, an additional group of healthy subject testing will be performed to validate the combined imaging and control systems and to construct a set of healthy data for use in clinical trials. Once clinical trials have proven the effectiveness of the system, commercialization of the artificial limb control system will progress.

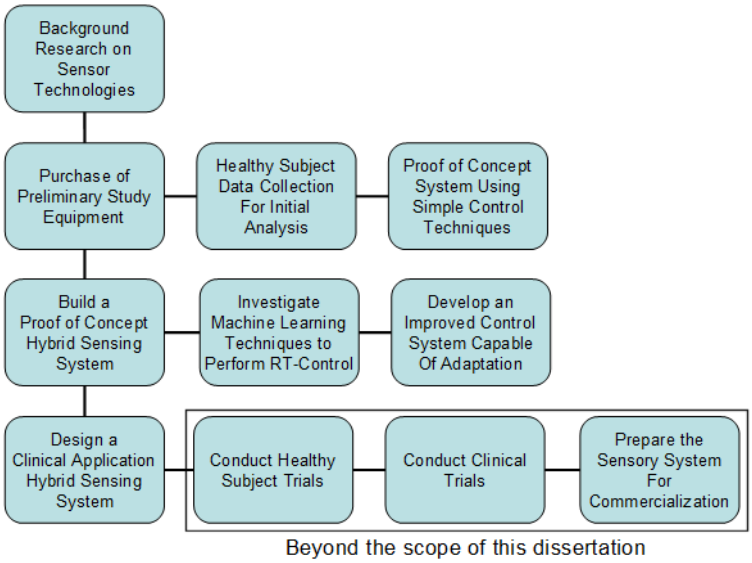


Figure 1.2: Planned Research Flow Chart

To complete these tasks, EEG, NIRS, and EMG are used for data acquisition. A simple control system is completed using a transfer function and a case matching algorithm to directly link input frequencies with output strength, machine learning techniques including k-nearest neighbor and neural networks are investigated, multiple imaging systems are designed and constructed, and a proof of concept artificial limb is fabricated. In order to prepare the system for final testing, a control system is prepared which uses principle component analysis (PCA) and non-linear neural networks to perform input classification in order to sufficiently predict the desired output, a database is created to store trained networks, from which a nearest neighbor algorithm may select

an appropriate network based on system identification generated through PCA. A final headset design has been prepared and is currently under construction.

CHAPTER TWO: LITERATURE REVIEW

Prior Research in Artificial Limb Control

While the field of lower extremity prosthesis has seen significant improvement, commercially available upper extremity solutions have remained stagnant for decades. Current research in this area generally focuses on high degrees of control, requiring invasive and expensive methods of generating control signals (Defense Sciences Office 2010). At the onset of this project we conducted a review of the state of the art in this area and have summarized our findings below.

Castillo et al. studied integration of sensory and motor mapping in a comprehensive magnetoencephalography (MEG) protocol. Their research does not discuss near-infrared spectroscopy or control methods for an artificial limb (Castillo 2003). Darlot et al. utilized monitoring of the central nervous system but have not expanded their research to artificial limb control (Darlot 2002, 379-394). Deecke et al. discuss the mapping of motor cortex functionality through the use of direct current electroencephalography (DC-EEG), MEG, single-photon emission computed tomography (SPECT), and functional magnetic resonance imaging (fMRI), but did not include discussion of the control of an artificial limb (Deecke 1996, 295-311). Donchin et al. have focused their effort on the analysis of how the brain controls motor function. The brain's ability to correct for errors in intended trajectory is discussed. Their research results will be useful in our proposed study when the user would like to move the limb in

a direction with high resistance (Donchin 2003, 9032-9045). Englehart and Hudgins have investigated the control of upper extremity prosthesis through the monitoring of myoelectric signals (Englehart 2003, 848-54). Eskiizmirliler et al. discuss a motor control scheme based on sensory-motor interaction modalities within the central nervous system, and control of a single joint limb segment through the use of two pneumatic McKibben muscles. This research is based on data collected from nerves and does not include a direct brain control (Eskiizmirliler 2001, 865-8). The Gale Group discusses an artificial hand that is capable of revolving its arm right and left, bending and stretching its wrist, and opening and closing its fingers. Their control scheme is based on the electric voltages transmitted by nerves. It uses a neural network to learn correlation between electric signals and control of the three ultrasonic motors that it uses to perform the three degrees of freedom listed above. This system utilizes the electric signals that are carried by the body's nervous system to control their robotic hand (Gale Group 1995). Guoqiang et al. have demonstrated the use of near-infrared spectroscopy in human muscle tissue and rat brain for the purpose of measuring metabolic activity (Guoqiang 2003, 164-74). Hoshino et al. use near-infrared spectroscopy of the brain to determine changes in blood oxygenation and deoxygentation during direct stimulation of the human brain using an electrode (Hoshino 2005, 272-275). Muller et al. describe the use of magnetoencephalography and functional magnetic resonance imaging. While EEG monitors electrical impulses in the brain, fMRI uses blood oxygen level dependent (BOLD) contrast to image the brain, which has a clear relationship with the blood oxygenation level data collected with near-infrared spectroscopy (Muller 2005, 109-16).

While many similar research projects have been conducted, we have not found any completed research projects that aim to use the same techniques as our project to accomplish our goals. Even if a similar project is underway, we hope to develop a sophisticated control algorithm using this combination of brain and limb monitoring techniques to create a better non-invasive control method than what is currently available.

Biological Imaging Techniques

Understanding the function of the human brain is always challenging. The brain is the most complex organ in the human body and is the central processing unit for human behavior, cognition, learning, memory, emotion, sensory, motor functions, etc. The basic building block for the human brain is the neuron, a specialized cell designed to transmit information to other nerve cells, muscle or gland cells (Carey 2008). To understand the complex function of the brain many imaging techniques have been developed (Webb 2003). The following summarizes some of these techniques.

Electroencephalography (EEG): EEG observes a small amount of electric current passing through the neurons during their activity. These tiny currents are detected and recorded by placing electrodes on specific regions of the brain. By using protocols based on stimulus and response, and by analyzing the recorded data, the functioning of the brain can be detected.

Positron Emission Tomography (PET): PET is one of the most important techniques for measuring blood flow or energy consumption in the brain. This method is based on the detection of radioactivity which is emitted when the positively charged

particles undergo decay in the brain. Three dimensional PET images, which reflect the blood flow, metabolic and chemical activities of the brain, are produced by the injection of substances containing positron emitting nucleotides. PET has been very useful for scientists in understanding effects of drugs on the human brain, and for learning what happens during certain brain disorders such as Parkinson's disease or strokes.

Magnetoencephalography (MEG): MEG is a recently developed technique that reveals the source of weak magnetic fields emitted by neurons. In this technique, an array of cylindrical shaped sensors are used for monitoring the magnetic field patterns near the brain, which are used to determine positions and strengths of activity in different regions of the brain. In contrast with the other imaging techniques, MEG can characterize the rapidly changing patterns of the neural activity, with millisecond resolution and can provide a quantitative measure of the strength of this activity in individual subjects. Moreover, by presenting stimuli at various rates, scientists can determine how long neural activation is sustained in the diverse brain areas that respond to such excitations.

Magnetic Resonance Imaging (MRI): MRI is a high-quality, three-dimensional image of organs and structures inside the body without X-rays or other radiation. MRIs are unsurpassed in focusing on anatomical detail and may reveal minute changes that occur over time. MRIs indicate when structural abnormalities would first appear in the course of a disease, how they effect subsequent development, and precisely how their progression correlates with mental and emotional aspects of a disorder. Tissues that contain a lot of water and fat produce a bright image in MRI, whereas those tissues that

contain little or no water, such as bone, appear black. MRI images can be constructed in any plane, and the technique is particularly valuable in studying the brain.

Magnetic Resonance Spectroscopy (MRS): MRS is a noninvasive technique related to MRI. It uses the same equipment but instead of blood flow, it measures the concentration of specific chemicals, such as neurotransmitters, in different parts of the brain. MRS also holds great promise by measuring the molecular and metabolic changes that occur in the brain. This technique has already provided new information about brain activities, such as brain aging, Alzheimer's disease, schizophrenia, autism, and stroke.

Functional Magnetic Resonance Imaging (fMRI): fMRI is one of the most advanced and useful techniques of brain imaging. This modality is based on MRI (Carey 2008, Webb 2003, Villringer 1997, Heeger 2002), which focuses on the radio signal emissions of chemical elements within a magnetic field. The protons of the hydrogen atoms, present in water or fat in the tissues are normally aligned randomly in different directions. But in the presence of a very strong magnetic field (many times the strength of earth's magnetic field) in an MRI machine, they align themselves in parallel like rows of tiny bar magnets. If the hydrogen atoms are knocked out of their alignment by a strong pulse of radio wave, they produce a detectable radio wave when they fall back into alignment. The emitted radio waves can be detected by the magnetic coils, the output of which can be converted into an image by a computer, based on the properties of different types of body tissues. fMRI is an extension of MRI, which is used for measuring brain activities during rest or activated conditions. This technique utilizes the paramagnetic nature of deoxy-hemoglobin, which is the basis of the BOLD signal detected in fMRI. By

using fMRI, researches have been able to produce detailed maps of brain areas underlying human mental activities in health and disease. This technique has also found its application in studies of the functions of the human brain ranging from primary sensory responses to the cognitive activities.

Electromyography (EMG): Most of the motorized prosthetic limbs currently being sold commercially are controlled by EMG and are referred to as myoelectric prostheses. EMG monitors the electric potential of flexor and extensor muscles in the remaining portion of the limb and based on the differential between these, it can be determined whether to close or open a prosthetic hand. This system requires the user to consciously flex muscles in order to control the artificial hand since the activity of remaining muscles would have normally controlled a different movement within the limb than the output of the prosthesis.

Functional Near-Infrared Imaging (fNIR): fNIR technique is based on the observation that brain activity induces changes in the optical properties of the brain. The NIR imaging is made possible due to the existence of an optical window in the near infrared laser wavelength range of 700-1000nm, where there is relatively little absorption of light by the biological tissues. Furthermore, oxy-hemoglobin and deoxy-hemoglobin, which are important indicators of biological activity in the tissue, are the chief absorbers in this region (Heeger 2002). This absorption indicates changes in the biological activity of the tissues and the concentration of the chromophores, specifically oxy-hemoglobin and deoxy-hemoglobin changes. These changes are in turn reflected in the amplitude or phase changes of the transmitted/reflected light.

The above mentioned functional imaging techniques have been used extensively and each of them have contributed to the understanding of the functions and internal mechanisms of the human brain. Each of these methods has unique capabilities, advantages and disadvantages and in many instances they are used in combination to complement and produce a better result than a single technique could. Based on the existing literature, EEG and MEG can monitor neural activity with a very good temporal resolution (100Hz to 1 KHz), but they have a poor spatial resolution; PET can measure blood flow, volume, and glucose metabolism; and fMRI can measure relative hemodynamic, metabolic and neuron signals. Compared to PET and fMRI, fNIR has poor spatial resolution and depth penetration but good temporal resolution (Webb 2003, Villringer 1997, Heeger 2002). On the other hand near-infrared spectroscopy (NIRS) offers biochemical specificity by being able to measure the concentrations of oxy-hemoglobin (oxy-Hb), and deoxy-hemoglobin (deoxy-Hb), whereas fMRI, PET, EEG, or MEG can measure either neuronal activity (EEG, MEG) or the vascular response to it (PET, fMRI). Finally the instrumentation for EEG and fNIR are portable and relatively inexpensive, compared to the other functional imaging techniques, creating a potential for their wide application in the field of human brain studies.

Recent studies performed utilizing brain monitoring techniques are developing brain computer interfaces (BCIs) to control everything from a mouse cursor, to a thought based game (Emotiv 2008), to an artificial limb. This area of research is very similar to our own and aims to develop direct brain control methods for many applications as well as artificial limbs. One of the most common sensing techniques being used is EEG, which

is an inexpensive and non-invasive method for continuous brain monitoring. Guger Technologies (g.tec) and Hitachi-Medical are two of the leading groups for the development of sensors for direct brain monitoring for medical purposes. G.tec uses EEG for direct brain monitoring while Hitachi-Medical utilizes near-infrared imaging (Guger Technologies 2008, Hitachi-Medical 2008).

Near-infrared imaging and EEG are both currently being improved upon to overcome some limitations such as sensing through hair and requiring reliable contact with the scalp to avoid serious errors in sensory feedback. Both technologies show a potential for use in artificial limb control as well as direct brain control for a multitude of other applications in the very near future.

Anatomy of Brian Command and Control

Understanding how the brain functions, and particularly how the brain interacts with the nervous system in order to control muscles and detect somatosensory information, is key in utilizing a brain imaging system for real-time control. The layout of the motor cortices and how they behave when active must be established to properly interpret data collected, and to correctly position sensing equipment for detection of specific activities.

The human brain is a complex organ that is not fully understood. Research in this area often requires risk to the patient, and/or significant costs, and the data collected can often vary significantly between subjects. Developing a low cost, low risk device to collect large quantities of data on neural motor commands could significantly help

advance this field of study. The currently accepted organization of the motor control sections of the brain and the properties of hemodynamic response are examined in this section.

Motor Cortex Layout and Complexity

The human brain controls all of the body's muscles through the nervous system. Subconscious control over various portions of the body is accomplished by the lower motor neurons in the brainstem, including heart rate, automatic breathing, balance control, and reflexive muscle control. More complex voluntary movements, however, are controlled by the upper motor neurons located in the motor cortices (Purves 2001). These cortices can be further divided into primary motor and premotor cortices.

The premotor cortex processes sensory data to prepare for motor activation, while the motor cortex carries out the actual commands to be sent to muscles. The relatively low level of electrical stimulus in the brain, that is needed to evoke a muscle response, demonstrates the directness of muscle control performed in the primary motor cortex (Purves 2001). Mapping between regions of the brain and corresponding muscle activation has been studied in animals as well as humans to better understand a variety of neural problems such as seizures and amyotrophic lateral sclerosis (ALS). These studies have shown muscle control is distributed laterally across the surface of the primary motor cortex with greater area assigned to portions of the body requiring more complex motor control such as the hands and face. It is also significant that the brain does not control a single muscle with a single region of the brain. A single point of activity in an upper

motor neuron will generally fire a combination of lower motor neurons, which results in a combination of muscle groups activating to accomplish an intended body motion (Purves 2001). This is most significant in control of the hands since they have a much higher dexterity than larger limbs with lower degrees of freedom.

The amount of force generated by a muscle is controlled by the frequency with which upper motor neurons are fired as well as recruitment of additional motor units. This allows approximation of intended muscle contraction based on the degree of activity in the brain. Motion in directions other than the basic flexion and extension of limbs will activate multiple upper motor neurons with varying degrees of intensity to produce the desired net result, with activation intensity changing through the course of motion as muscle activity is more and less directly correlated with the neurons involved (Purves 2001).

The premotor cortex is further broken down into the lateral and medial premotor cortices. The lateral premotor cortex interprets sensory data and pre-processes intended muscle actions well before the primary motor cortex sends commands out. This cortex does show a considerable amount of direct correlation between activation and muscle activity, but it handles a variety of additional variables for things such as conditional movement, movement intent, and observation of another being's movement (Purves 2001, Saygin 2004). Lesions in this portion of the brain tend to lead to a patient's inability to respond to visual or verbal cues to initiate a motion, even if they are fully capable of performing the given motion and understanding the significance of the cue (Purves 2001).

The medial premotor cortex handles preparation for muscle activities that are in response to internal cues. Memorization of planned muscle sequences, as well as more spontaneous motions, is handled within this cortex. This region is of particular interest for direct brain control of artificial limbs, since self-initiated motor control is often visible within this cortex one to two seconds prior to muscle activation.

Hemodynamic Response

Monitoring the volume and characteristics of blood within the motor cortices is the basis for fMRI as well as fNIR. The characteristics of blood flow within the brain, and the time for a neural response to be fully achieved, pose significant challenges for real-time control using this method.

After an increase in neural activity in the motor cortex, the local blood flow increases more than the metabolic rate of oxygen in the same region. This flow of oxygenated blood into the area of activation changes the ratio of oxygenated blood to deoxygenated blood, which can be used to indicate an activated neuron (Buxton 2004, 220-33).

Some of the significant factors that need to be considered when using a BOLD signal include:

- Blood flow increases much more than the metabolic rate at the point of activation.
- Blood flow and BOLD responses to any magnitude of signal display a one to two second delay before significant changes occur.
- Changes in blood flow and oxygenation typically last four to six seconds for an impulse input.

- Sustained stimulus typically will result in a level increased signal for much of the duration of the sustained muscle activity with possibilities of overshoot at the start and end of the signal.

It is also typical for the BOLD signal to take longer than thirty seconds to fully return to prior levels after a large or sustained activity. BOLD signals have also indicated an initial dip immediately after activation of a muscle, lasting for the one to two seconds prior to the normal increase resulting from the activation (Yacoub 2001). This is potentially the result of a metabolic rate increase at the site of activation prior to the inflow of oxygenated blood. There are also significant concerns regarding the non-linearity of BOLD signals in response to varying types of inputs as well as discrepancies in methods to remove baseline blood flow variations from basic physiological phenomena such as heart rate, level of activity, and movement of the head. The actual mechanism that couples neural activity to blood flow and oxygenation is not fully agreed upon and is an ongoing area of research (Buxton 2004, 220-33).

The use of animals as test subjects allows for a more precise and higher resolution view into the hemodynamic response of the brain to neural activity. Since the motor cortex also processes somatosensory data, stimulating a portion of the body can act as a reliable input to the system through biological means without large amounts of patient variation. Stimulating the whiskers of rats, while an imaging probe was inserted into the area of the brain that corresponded with the input signal, provided researchers with a very large sample population in a more invasive manner than would be possible with human subjects. Their results show an increase in deoxygenated hemoglobin with moderate intensity almost immediately after stimulus, followed by an increase in blood volume and

a very large increase in oxygenated hemoglobin beginning one second after stimulus and lasting for at least three seconds (Devor 2005).

The tests performed by Devor et al. also indicate a significantly long response time to a ramp input. This is mostly the result of the substantial increase in blood flow into an area that is continuously being activated causing a longer settling time (around ten seconds). For control purposes it does not present an unusable signal, but will require an algorithm that can handle long duration activations in which muscle activation no longer corresponds with a variation in blood oxygenation, but instead is determined by the level of raised state relative to surrounding inactivated regions.

Another concern for real-time control using BOLD signals is the behavior observed when multiple stimuli occur within a short time period, which is required for a replacement limb to feel natural. Results indicate that stimulation in inter-stimulus intervals shorter than six seconds result in a non-additive total signal. Subtracting the single stimulus results from the total signal of a multiple stimulus test indicate that the second stimulus produces an attenuated and delayed signal (Huettel 2000, 547-53). This poses a risk for artificial limb performance degradation as degree of complexity and repetition of movement increase.

It is also important to account for potential variation between subjects in brain response characteristics. In another study by Huettel et al., the variations that arise from the age of the patient were analyzed. Their tests indicated that older patients had a BOLD signal response time that was faster than younger patients, but also had a higher degree of variance, fewer points of activation, and a substantially lower signal to noise ratio

(Huettel 2001, 161-175). This variance typically did not affect the peak signal strength, the time of the peak signal, or the time to return to normal levels after activation, but does indicate that more significant training may be required for older subjects.

The above characteristics make artificial limb control using a BOLD signal a challenge, as well as a very dynamic indicator of brain activity. To further understand the relationship between neural activity, cerebral blood characteristics, and muscle activation, all of these aspects must be taken into account to avoid erroneous conclusions. Through the use of machine learning and adaptation, developing a control system to handle this complex set of inputs may achieve the goals of this research.

Machine Learning Techniques

A variety of different methods have been developed to use computers to find patterns in data for the purpose of classification or regression. Selecting a technique that is appropriate for the target data set is the most important step in solving a complicated problem using one of these methods. A brief description of available techniques that may be suitable for our application is provided in this section.

Artificial Neural Networks: A family of mathematical models that depend on the interconnectivity of neurons within a network. They are loosely based on the structure of biological neural networks and aim to produce the same learning by association that is present in these systems (Heaton 2008). There are a variety of techniques that have been developed to train the network to correlate input and output information, the most basic of which is back propagation (Heaton 2008). Artificial neural networks do however

require a great deal of computational resources for complex problems and require a diverse set of training data in order to function in many real-world applications (Pomerleau 1993).

Support Vector Machines: SVMs build a hyperplane or set of hyperplanes between outcomes in a system, defined as the dividing space that is farthest from any sample point. By establishing the location of these planes in a high-dimension space, the outcome of a new set of inputs can be determined by which side of a plane it is located on (Vapnik 1995). Several types of SVMs have been developed to improve the effectiveness of this technique, but the main drawback of this technique is still that it is designed to solve two-class tasks, which requires multi-class problems to factor the outcomes into several recurring two-class problems (Duan 2005).

Logistic Regression: Using linear or binomial regression, logistic regression calculates the probability of an outcome based on the combination of input variables. In order to not overestimate odds of an event, this method requires a very large sample set in order to reduce this effect to near-true values (Nemes 2009).

Naive Bayes: Using supervised learning, the naive Bayes classifier determines the probability of an output based on the combination of the probability of that output for each input independently. By not considering the covariance of input data, the training set required to produce classification information is relatively small (Zhang 2004).

k-Nearest Neighbor: A type of memory-based learning, nearest neighbor algorithms simply find the closest matching set of inputs in a database of known input-to-output recordings. k-Nearest Neighbor expands on this technique to find the k-closest

matches and uses a combination of their input-to-output information to make a better informed determination of the membership of the new input data. This technique uses lazy learning and does not perform any computation until run-time, making it a computationally demanding technique for very large sets of known-output data (Dasarathy 1994). While weighting the contribution of neighbors, based on the inverse of their distance to the new point, can help interpolate the output for cases that do not closely resemble member data, it may have difficulty predicting the classification of data that is dissimilar to previously seen inputs.

Decision Trees: There are several different types of decision trees such as bagged trees, boosted trees, boosted stumps, and random forests. A brief look into these indicates the random forests method is the most likely potential solution from this family.

Random Forests: Using a combination of many decision trees, this technique makes a classification based on the most common output of the collection of trees. The decision trees are constructed using random combinations of variables, then trained based on a known set of outputs and are not pruned (Ho 1998). Over-fitting is a concern when using noisy data sets with random forests (Segal 2004).

A 2006 paper by Caruana evaluated the effectiveness of various machine learning techniques across a variety of binary classification data sets. Their study determined that prior to calibration, bagged trees, random forests, and neural networks proved the best performance across the problems they were tested upon. After calibration, boosted trees produced the best results and SVMs achieved the performance of neural networks and nearly performed as well as boosted trees, random forests, and bagged trees (Caruana

2006). Since it was not our intention to perform calibration on the data set and our data tends to include inconsistencies, neural networks are the solution that appears to be most likely to succeed using the data produced by this study. For the purpose of not over-complicating the problem, linear solutions will be explored first to rule out a simple solution. To explore how well over-fitting can solve the brain-to-muscle mapping problem, a k-nearest neighbors algorithm will be explored as well.

CHAPTER THREE: DESIGN OF HEADSET

Requirements

The sensory system used to monitor brain activity must meet the following minimum specifications:

- The device shall conform to the size and shape of many individuals' heads (sensors must be able to adjust to be normal to the surface and fit head widths ranging from five to seven inches wide).
- The device shall be comfortable enough to wear for extended periods of time (at least one hour).
- The device shall have an array of surface EEG electrodes capable of monitoring electrical activity of the brain across the relevant portions of the primary motor cortex.
- The device shall have an array of NIRS light sources and detectors capable of monitoring general hemodynamic behavior occurring across the relevant portions of the primary motor cortex.
- The data acquisition system shall be able to operate continuously for at least one hour at a time.
- The data acquisition system shall have a user interface capable of performing data acquisition, data analysis, control system training, and system identification.
- The control unit shall perform data analysis and be capable of identifying upper-extremity actions being performed as well as the magnitude of that activity, based on EEG and NIRS sensor data being collected in real-time.
- Classification of the output activity shall meet or exceed an accuracy of 80%.
- Estimation of the output activity intensity shall be within 20% from measured values obtained using EMG.

Preliminary Trials Using Commercially Available Equipment

Two clinical trials were performed to determine the usefulness of NIRS and EEG sensory techniques in this application. The first set of trials was performed to determine the effectiveness of NIRS for monitoring brain activity across the motor cortex and verify those readings against magnetoencephalography data collected simultaneously. The second set of trials was performed to collect EEG and EMG data simultaneously to experimentally determine the correlation between electrical activity in the brain and muscle activation.

Through a proof of concept experiment, a set of LEDs and photodiodes were assembled into a headband and were put on six human subjects with their heads shaved to facilitate light transmission and collection (data sheet in Appendix A.1). These subjects were asked to perform several simple movements while photodiode sensor data was being collected. During each activity, the general tendency was for oxygenated blood concentration to decrease briefly, followed by an increase in total blood volume in areas of the brain that were activated by the movement. Multiple patients performing the same muscle activity produced similar response in blood oxygenation and de-oxygenation concentration through the motor cortex. Indication of brain activity using NIRS matched with MEG data collected.

The time between muscle activation and recognizable brain activity using NIRS was generally one to two seconds, indicating that the use of NIRS as the primary indicator for brain activity may be insufficient for real-time control of an artificial limb.

For this reason NIRS has been reallocated as a feedback element in the overall system diagram and will be used for system error mitigation, adaptation, and system identification in future studies.

Seventeen volunteers participated in a healthy subject trial to collect EEG and EMG data during various elbow and wrist activities (data sheets in Appendix A.2 and A.3). EEG sensors were placed according to the international 10-20 system for sensor placement, with active sensors located near C3 and C4 positions to emphasize sensitivity to upper extremity movement (Immrama Institute 2010). EMG sensors were placed on both arms over the major muscle groups associated with elbow and wrist flexion and extension. After improvements were made to data collection techniques based on the first two sessions, fourteen of the fifteen remaining trials produced results that indicated reliable connections were established between the sensors and the scalp. During the session that is excluded from the data, the level of background noise present on the signal never subsided to useable levels. Subjects performed elbow and wrist flexions and extensions at five-second intervals for each arm separately as well as a simultaneous activation of both arms. Following each activity, another session of data was collected were the subject visualized repeating the same activity while not moving the limb. During the visualization sessions, less than half of the subjects displayed a response in the brain similar to the activity that they produced during physical movement of the limb and the response had much smaller amplitude.

Through visual inspection during data collection, a significant increase in Alpha wavelengths (8-12 Hertz) was observed during elbow flexion, while a significant increase

in Delta wavelengths (1-3 Hertz) was observed during wrist flexion. Distinction could also be made between a right arm movement and a left arm movement by observing the difference between the right and left channels. If a given activity caused an increase in the differential between the channels during activation of the right side of the body, it would have a corresponding decrease in differential during activation of the left side of the body. One concern using this distinction was the lack of a unique characteristic for when both limbs are moved simultaneously, which resulted in the direction of the change to be small and likely dependant on the quality of the connection to the scalp for the two sensors.

These results indicate that the Fast Fourier Transform (FFT) of data can be used as an indicator for muscle activity, with specific interest in frequencies one through fifteen Hertz. However a more advanced data analysis will need to be performed on the data to accurately discern the activity being performed. This more complex analysis is necessary because some subjects have demonstrated being much better indicators for muscle activation across all of the activities performed, while others didn't produce the correct markers. For this purpose we have proposed a learning method to build the correlation between the multiple inputs and multiple outputs of this system.

Sensor Selection and Use

To improve upon the sensory systems used during the preliminary trials, a new sensor system was designed and fabricated. The analysis of each sensor and the selection

of appropriate parts are detailed in this section. Results from using these sensors are presented in the subsequent sections on the various sensor systems.

NIRS

The NIRS device used in the preliminary studies utilizes LEDs that combine multiple light emitting elements into a single casing, with six diodes of each wavelength (730nm and 850nm). However, in general use the optical casing used to house these LEDs (flat glass can) causes too much light to be lost as the result of hair. For this reason several attempts were made to use optical fiber to bypass this interference (Aasted 2008).

Upon determining that light transmission using the original optics would not satisfy the requirements of this project, new LEDs were investigated to overcome the LED housing issue. The original diodes typically operated with a forward current of 40mA and had the capacity to operate at up to 110mA. From the specifications sheet for these LEDs (Appendix A.4), the optical power when using a forward current of 50mA is 50mW for 730nm and 60mW for 850nm. Similar LEDs by the same manufacturer only have one element for each wavelength in a combined housing, allowing for the use of a glass ball lens type case (data sheet available in Appendix A.5). The two types of LEDs operate at slightly different wavelengths, so the constants used to calculate blood composition have to be adjusted, but the frequencies used in the new LEDs (735nm and 850nm) are still valid for this application. The new LEDs are rated for the same forward current as the originals, but only produce 9mW of optical power at each wavelength using 50mA of current. To match the power output of the original LEDs without

modifying the current supply, six LEDs must be active in place of one. While the LEDs have the capacity to pulse at substantially higher power levels, the bulbs can become warm with extended use and the wires used to transmit power to the LEDs would have to be enlarged in order to carry the additional current. The circuit design uses parallel connections between similar frequency leads, which results in a very high current demand when all of the lights are used simultaneously. Fortunately, the device used in the preliminary study was intended for a slightly different application and was designed to monitor a wider area, so additional light sources and fewer sensors was a practical solution (sensor data sheet available in Appendix A.6).

The first generation combined sensor device was designed to use four, eight, or sixteen of the ball-lens LEDs in a circular pattern for each sensor with a source to sensor spacing of one inch to provide the depth of penetration necessary to image the surface of the brain. This allowed the research team to evaluate the performance of the ball-lens type LED and make determinations regarding the interference still present when monitoring through hair. The third generation device also uses the ball lens type LEDs and groups them into sets of four, so that in addition to running all sixteen LEDs that surround a sensor, groups of eight or single sets of four LEDs can be activated and in doing so, increase the resolution of the imaging system. The lower resolution, sixteen LEDs per sensor option is retained in this design due to the losses observed while testing the first generation device. While sufficient light does generally reach the sensors when using eight sources, increasing to sixteen can overcome excessive blocking that can occur from hair. Decreasing the number of LEDs used in unison allows the system to increase

the resolution of the data being collected by isolating the areas between active lights and sensors. The resulting sampling frequencies are presented in Table 3.1.

Table 3.1: Sampling Frequencies for Different Devices and Configurations

Device Configuration	Sampling Frequency (Hz)
Preliminary Trials Headband	3
First and Second Generation Headsets	32
Third Generation Headset, Sets of 16	21.33
Third Generation Headset, Sets of 8	18.29
Third Generation Headset, Sets of 4	9.14

EEG

The EEG sensors used in the preliminary studies proved to be useful for collecting electrical brain activity during seated, upper-extremity activities. However, the system uses a ground point at the centerline of the head, which causes large disturbances in the data when the legs are moved. To counteract this problem a new set of references were used that do not use a ground point. This new design keeps one reference at a constant voltage and adjusts the other reference's voltage to maintain a steady current. The references in this system are positioned on far lateral sides of the motor cortex.

Additionally, the EEG sensors used in the preliminary studies and the first generation hybrid sensor system use cotton wrapped electrodes, which can become uncomfortable during extended use. To solve this problem a new type of electrolyte holding material was purchased, which has a cylindrical form to provide less pressure in the contact with the scalp. The third generation sensor system uses electrical contact pads that are more suited for use with the cylindrical form.

The EEG frequencies of interest in this study are one through fifteen Hertz, with potential interest in frequencies up to 100 Hz in future applications. To ensure that a

sufficient sampling rate is available, the first and second generation EEG systems operate on an interrupt driven sampling frequency of 128 Hz, with a low-pass filter. The frequency will be increased to 256 Hz in the third generation headset and while higher sampling frequencies are feasible, the computational requirements of the control system increase unnecessarily when calculating the FFT of larger data sets and in order to evaluate frequencies down to one Hertz, the last second of data must always be used.

EMG

The EMG sensors used in the preliminary trials proved to be suitable for continued use in later phases of this project. However, using the triode sensor pads, they are limited to measuring major muscle activities and are not well suited to monitoring hand actuation. For this reason a joint angle measurement system should be investigated if more complex activities are required in future studies. The EMG sensors have their own internal sampling rate and data processing, the output of which is the muscle activation intensities, which are sampled at the same frequency as the EEG system, 128 or 256 Hz, to ensure that an output value is present for every input reading during training.

Sensor Data Processing

Software was written in C++ to facilitate the use of the new hardware systems. Other than an included file to perform serial communication via the CSerial class, which is LGPL-licensed, every element of the basic data collection and processing was written

by the research team for the purpose of making adjustments to the system simple for future use of the hardware and software in control systems. The software for this project is written to operate in a Microsoft Windows Vista console interface, to enable use of the laptop dedicated to this project.

Every time a function of the software requests data from the headset system, the following tasks are performed to prepare the data. Several temporary variables are initialized that will be used to analyze the data being received and then a read command is used to received data from the serial port, store that data in the read buffer, and record how much data was received. The length is then used to repeat an operation that unpacks the serial stream into useful data. The data is then stored in a sample buffer and values for a "sample-in" pointer and "last sample" are set.

While the sample-in pointer is greater than the sample-out pointer, the routine to get new samples is repeated, which returns the 19 variables that were stored in the sample buffer and increments the sample-out pointer. This data is then sorted into EEG, EMG, NIRS, and NIRS status values and stored in a global structured variable. The EEG data is currently passed without modification into an EEG information variable, but two variables, eegData and eegInfo, exist distinctly from each other for the option to use the sum and difference of corresponding channels (Equations 3.1 – 3.4) instead of the raw channel values.

$$EI_0 = ED_0 - ED_1 \tag{3.1}$$

$$EI_1 = ED_0 + ED_1 \tag{3.2}$$

$$EI_2 = ED_2 - ED_3 \tag{3.3}$$

$$EI_3 = ED_2 + ED_3 \tag{3.4}$$

Where EI_n represents the information value to be used and ED_n represents the raw data from the EEG system.

NIRS data is sorted into 735nm, 850nm, or off-state and stored in the corresponding data arrays. The current blood oxygenation and blood volume is then calculated using the modified Beer-Lambert equation (Equations 3.5 – 3.8). When all of the data received from the serial communication is processed in this manner, the total number of complete sample sets that were handled is returned to the software to track the current time index and most recent values.

$$\Delta A_\lambda = \log_{10}\left(\frac{I_0(t)}{I_0(0)}\right) = \left(\sum_{t=1}^{2} \Delta c_i . \varepsilon_{\lambda t}\right).d.DPF \tag{3.5}$$

$$\begin{bmatrix} \Delta A_{\lambda 1} \\ \Delta A_{\lambda 2} \end{bmatrix} = \frac{1}{d} \begin{bmatrix} \frac{\varepsilon_{\lambda 1 \cdot oxyHb}}{DPF_{\lambda 1}} & \frac{\varepsilon_{\lambda 1 \cdot doxyHb}}{DPF_{\lambda 1}} \\ \frac{\varepsilon_{\lambda 2 \cdot oxyHb}}{DPF_{\lambda 2}} & \frac{\varepsilon_{\lambda 2 \cdot doxyHb}}{DPF_{\lambda 2}} \end{bmatrix} \begin{bmatrix} \Delta c_{oxyHb} \\ \Delta c_{doxyHb} \end{bmatrix} \tag{3.6}$$

$$oxy = \Delta C_{oxyHb} - \Delta C_{doxyHb} \tag{3.7}$$

$$BV = \Delta C_{oxyHb} + \Delta C_{doxyHb} \tag{3.8}$$

To utilize the Modified Beer-Lambert equations, the equations are solved using Equations 3.9 – 3.13.

$$\Delta A_{i,j} = \log\left(\frac{N_i}{B_i}\right) \tag{3.9}$$

Where $\Delta A_{i,j}$ is the light attenuation, N_i is the raw NIRS sensor data, and B_i is the baseline value. The subscript i indicates the sensor location and j indicates 735nm or 850nm wavelength.

$$\Delta c_{i,1} = \frac{\Delta A_{i,0} * \frac{\varepsilon_{850,Doxy}}{DPF_{850}} - \Delta A_{i,1} * \frac{\varepsilon_{735,Doxy}}{DPF_{735}}}{\frac{1}{D} * \left(\frac{\varepsilon_{735,Oxy}}{DPF_{735}} * \frac{\varepsilon_{850,Doxy}}{DPF_{850}} - \frac{\varepsilon_{735,Doxy}}{DPF_{735}} * \frac{\varepsilon_{850,Oxy}}{DPF_{850}} \right)} \tag{3.10}$$

$$\Delta c_{i,1} = \frac{\Delta A_{i,1} * \frac{\varepsilon_{735,Oxy}}{DPF_{735}} - \Delta A_{i,0} * \frac{\varepsilon_{850,Oxy}}{DPF_{850}}}{\frac{1}{D} * \left(\frac{\varepsilon_{735,Oxy}}{DPF_{735}} * \frac{\varepsilon_{850,Doxy}}{DPF_{850}} - \frac{\varepsilon_{735,Doxy}}{DPF_{735}} * \frac{\varepsilon_{850,Oxy}}{DPF_{850}} \right)} \tag{3.11}$$

Where $\Delta c_{i,j}$ is the change in chromophore concentration in micro-Moles (μM), $\varepsilon_{n,m}$ is the specific absorption, DPF_n is the differential path-length factor, and D is the sensor to source distance in centimeters.

$$oxy_i = \Delta c_{i,0} - \Delta c_{i,1} \tag{3.12}$$

$$BV_i = \Delta c_{i,0} + \Delta c_{i,1} \tag{3.13}$$

Which represent the oxygenation and blood volume for each sensor.

The data analysis software also has a function built in to switch from the normal call to get data each time a sample is taken from the headset, to get mirrored data, which functions the same as the original function, but switches channel information from the right and left sides of the head. This is intended to be used to train the system using a healthy side of the body and then control the system using the mirrored brain signals during use. This is based on the symmetry displayed in the lateral topography of the

motor cortex and was inspired by a possibility to treat phantom limb pain through using our brain monitoring system. Mirror therapy is currently used to help alleviate phantom limb pain by observing the healthy side of the body in a mirror, while performing repeated activities. If our system could be used to elicit healthy brain activity on the side of the brain causing the phantom limb pain, it could be much more therapeutic than visual feedback that requires actuation of the healthy side.

First Generation Sensor Configuration

To improve the capabilities of our brain monitoring system, we have designed and built a new sensory helmet, which incorporates four active EEG sensors and three NIRS sensors into a set of adjustable sensor modules (Figures 3.1 and 3.2). The combination of the brain's electrical activity with the hemo-dynamic response being observed should allow for a dynamic control system to be implemented that will account for errors and adapt to the user over time.

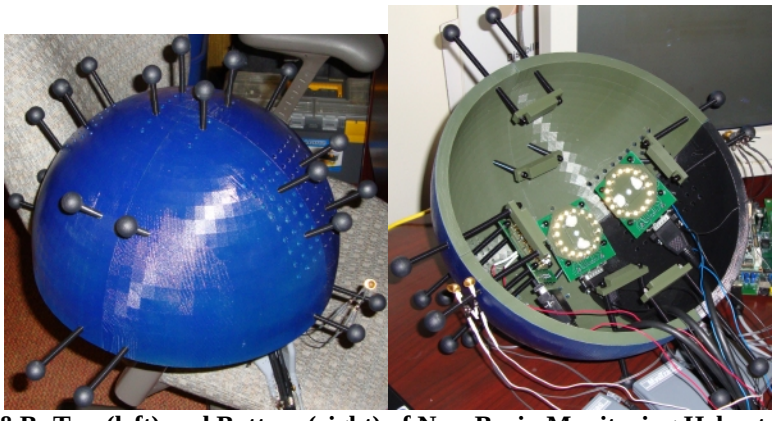


Figure 3.1 A&B: Top (left) and Bottom (right) of New Brain Monitoring Helmet, Showing the Adjustable Sensor Array.

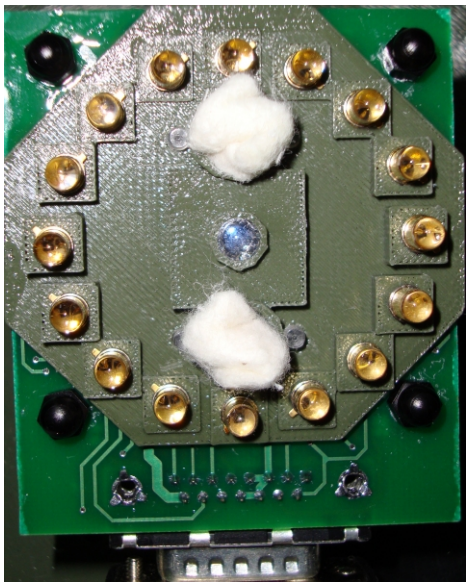


Figure 3.2: Sensor Module NIRS and EEG Component Layout.

Our initial tests of the new EEG sensor capabilities can be seen in Figures 3.3-3.6. These figures display the difference in signal across each of the two sensor pairs. The significant factor in determining the effectiveness of these sensors is the frequency composition of the signals and it is clearly visible that the amplitude of the signal is sufficient to provide that data.

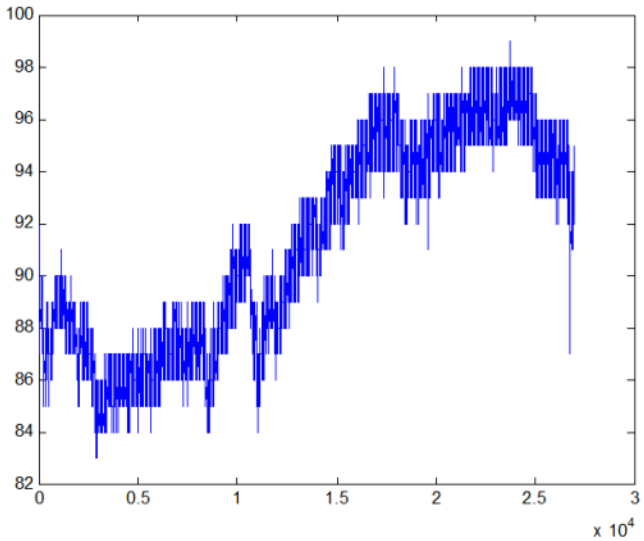


Figure 3.3: Difference of EEG Channels 1 and 2, Signal Magnitude vs. Sample Index

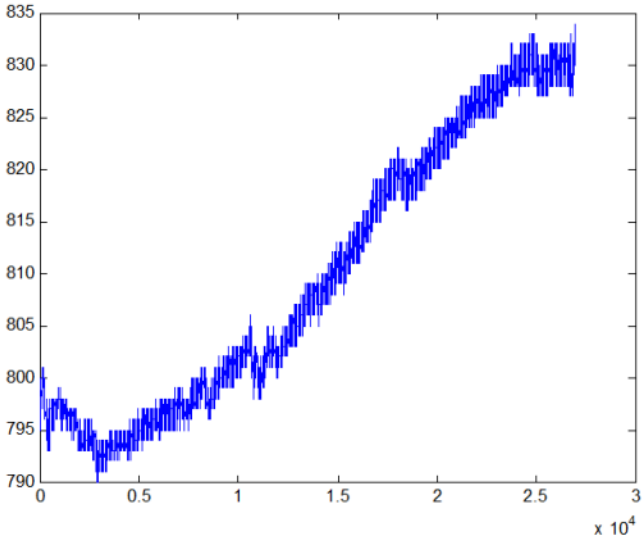


Figure 3.4: Summation of EEG Channels 1 and 2, Signal Magnitude vs. Sample Index

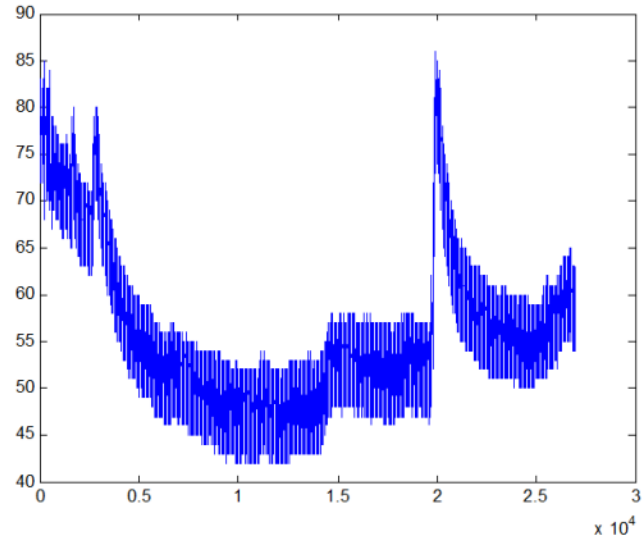


Figure 3.5: Difference of EEG Channels 3 and 4, Signal Magnitude vs. Sample Index

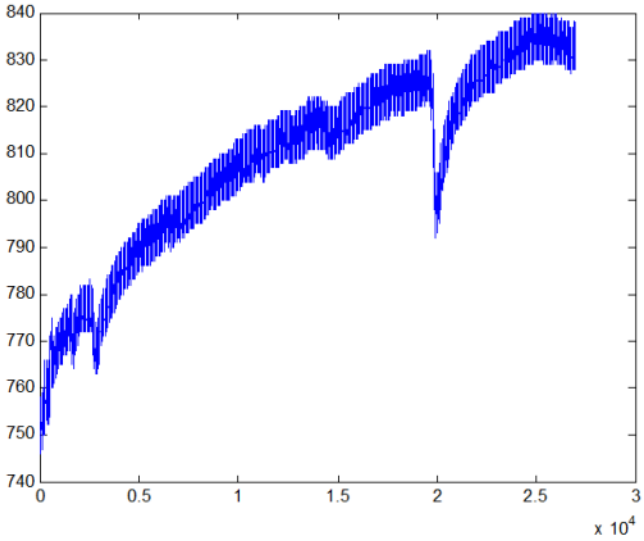


Figure 3.6: Summation of EEG Channels 3 and 4, Signal Magnitude vs. Sample Index

The EMG sensors being used to train the control system are identical to the ones used in the previous stages of this research, but for the purpose of demonstrating the capabilities of the current system, the EMG data recorded during the EEG session above is provided in Figures 3.7-3.10. These readings correspond with six repetitions each of bicep flexions, triceps extensions, wrist flexions, and wrist extensions. Good isolation is observed in most activities, however sensor placement and sensitivity can show activation when little to no flexion of the joint is present.

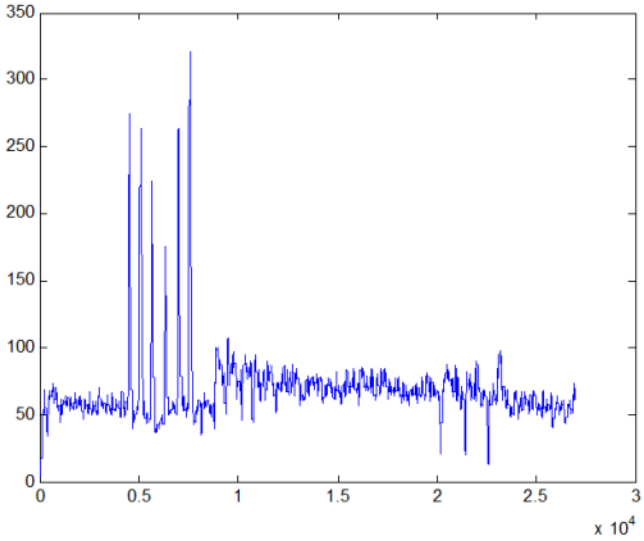


Figure 3.7: EMG Positioned Over the Bicep, Signal Magnitude vs. Sample Index

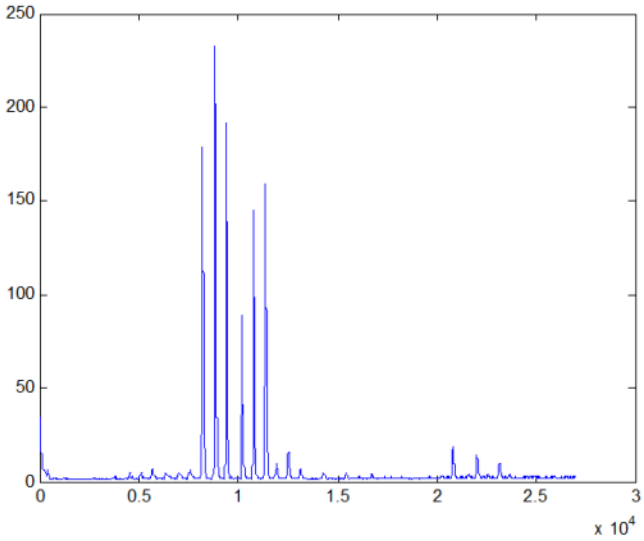


Figure 3.8: EMG Positioned Under the Triceps, Signal Magnitude vs. Sample Index

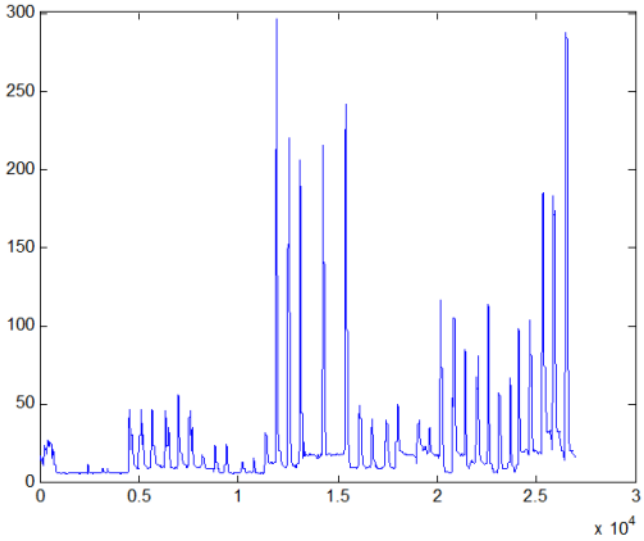


Figure 3.9: EMG Positioned Under the Forearm, Signal Magnitude vs. Sample Index

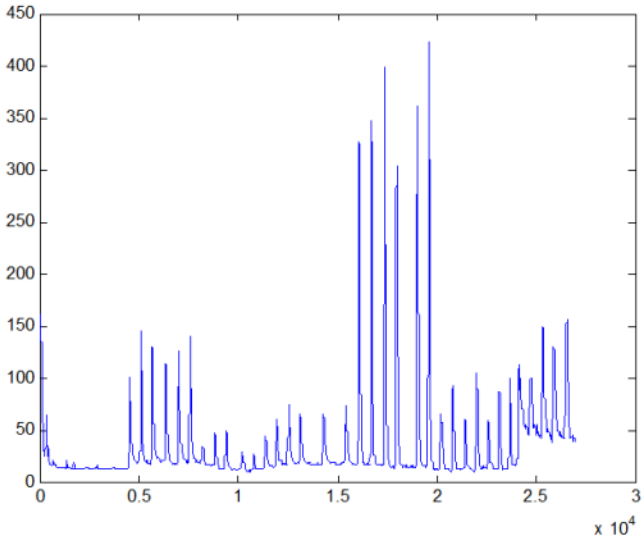


Figure 3.10: EMG Positioned Over the Forearm, Signal Magnitude vs. Sample Index

The most significant change to our brain monitoring system is the simultaneous inclusion of NIRS in our sensor capabilities. Figures 3.11 - 3.13 demonstrate the changes in light refraction during the activities displayed above. The data index is different for the NIRS because it is sampled at 32 Hz, compared to the sampling rate of 128 Hz used for both EEG and EMG data collection. The data represented below is the difference of the

intensity of light being refracted at 850nm and 735nm wavelengths, minus the baseline value collected for that sensor. These plots do not represent the actual blood volume and oxygenation levels.

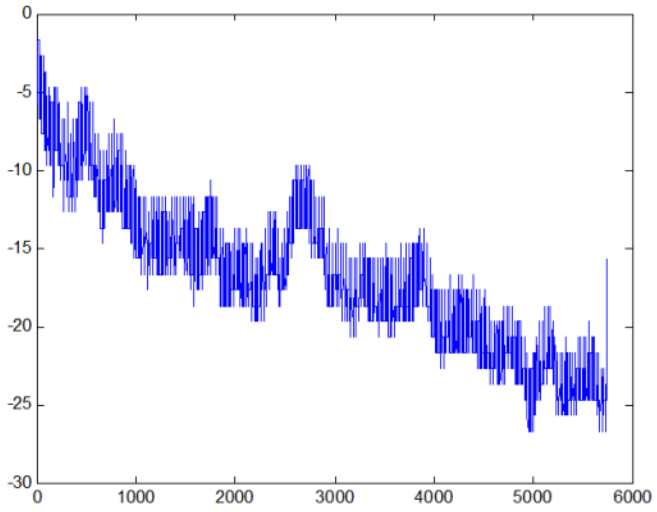


Figure 3.11: NIRS Channels 1 and 2, 850nm – 735nm, Left Side of Brain, Signal Magnitude vs. Sample Index

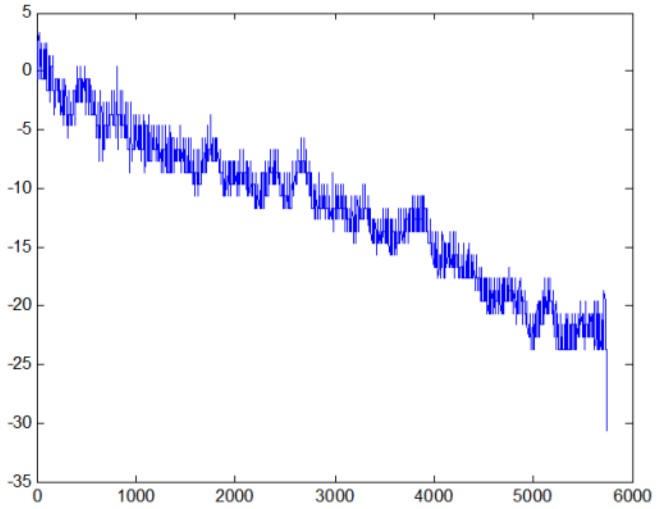


Figure 3.12: NIRS Channels 3 and 4, 850nm – 735nm, Center of Brain, Signal Magnitude vs. Sample Index

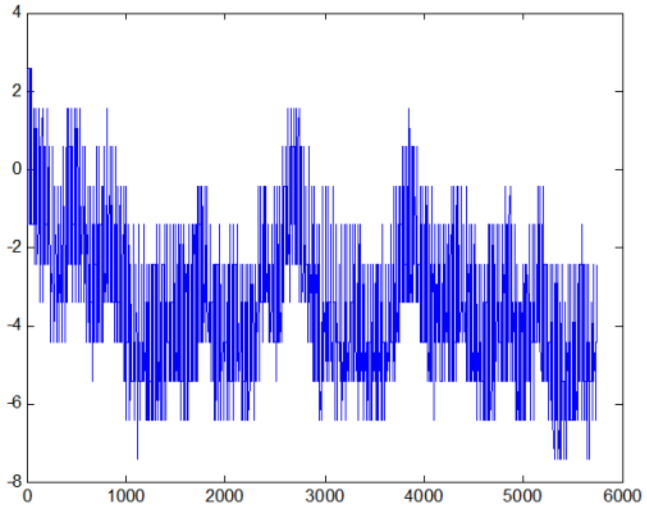


Figure 3.13: NIRS Channels 5 and 6, 850nm – 735nm, Right Side of Brain, Signal Magnitude vs. Sample Index

Second Generation Sensor Configuration

While the first generation helmet system met the sensor requirements for simple limb actuations, it was determined that additional sensors would be required for a side project investigating treating phantom limb pain using this system. In preparation for designing a new headset with additional sensors and a more flexible system for contouring to an individual's head shape, an intermediate headset was built and used for testing the control algorithms included in later chapters (Figures 3.14-3.16).

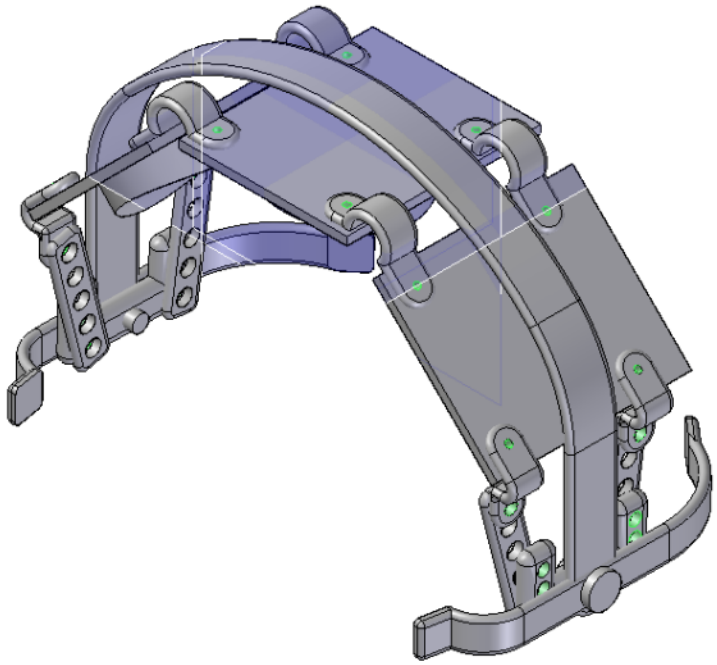


Figure 3.14: Headset Reconfigured to Work with First Generation, Rigid Sensor Boards.

Based on the results of using the headset with the old system, some of the parts were modified and final versions were created for trial or clinical application.

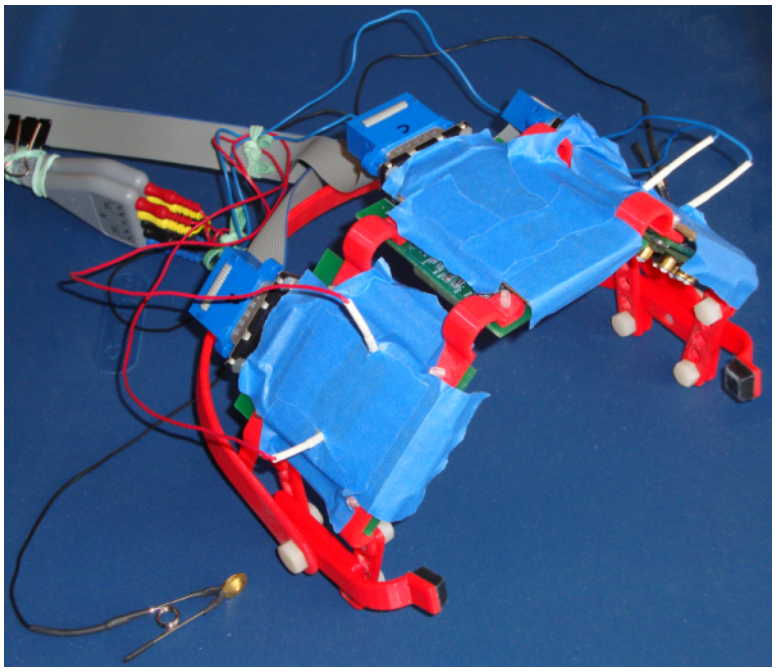


Figure 3.15: Intermediate Headset Finished Product

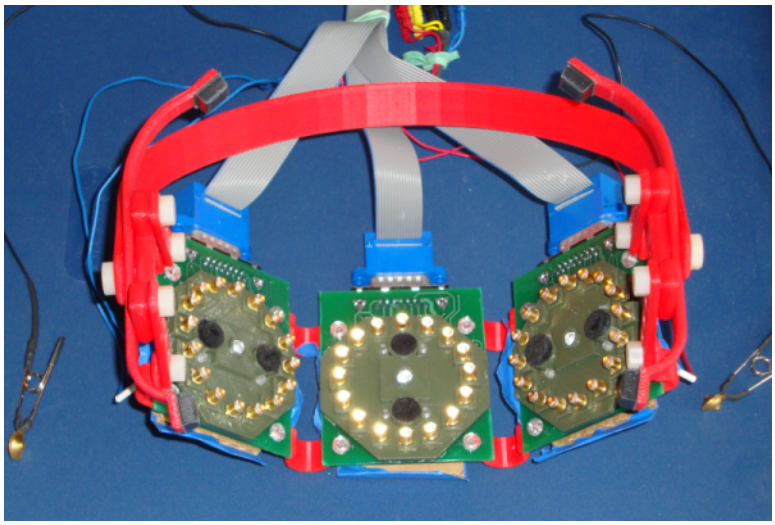


Figure 3.16: Sensor Array in Intermediate Headset

Third Generation Sensor Configuration

Based on the results of the intermediate "second generation" headset, the following designs for a third generation headset, with additional sensors and a flexible circuit board, have been finalized:

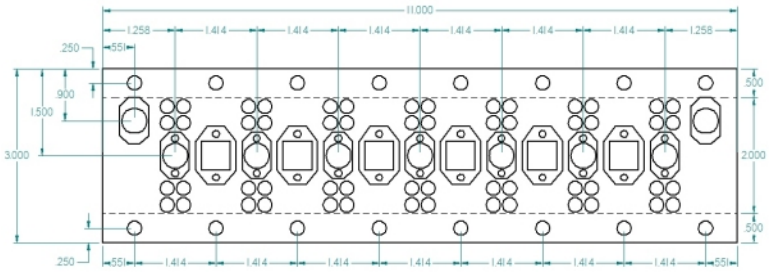


Figure 3.17: First Draft of Sensor Layout for 3rd Generation Headset

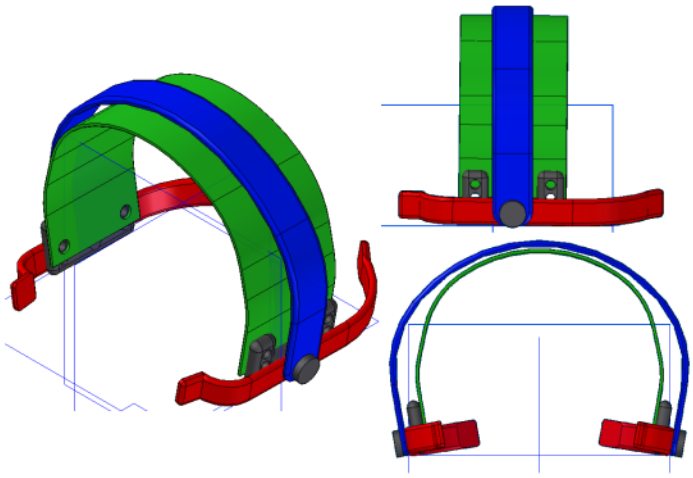


Figure 3.18: First draft of new headset design

Figure 3.19: Final Headband Design

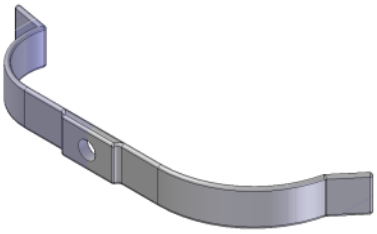


Figure 3.20: Final Side Support Design

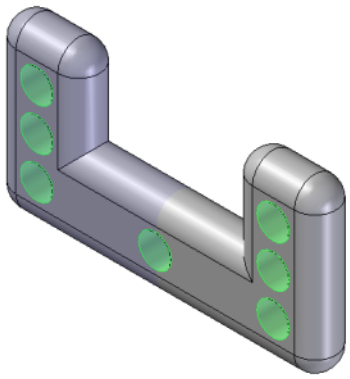


Figure 3.21: Final Mount Design

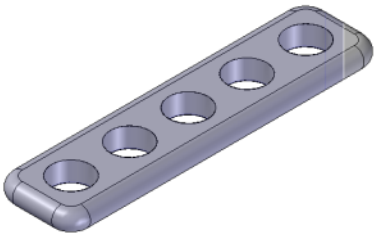


Figure 3.22: Addition of a Swing Arm Design

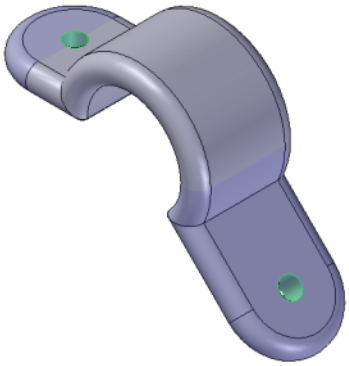


Figure 3.23: Backwards Compatible with First Generation Board Connector Design

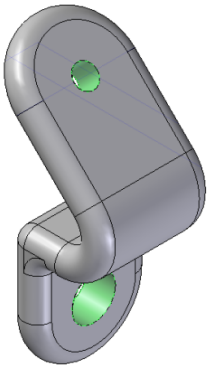


Figure 3.24: Backwards Compatible with First Generation Board Mount Design

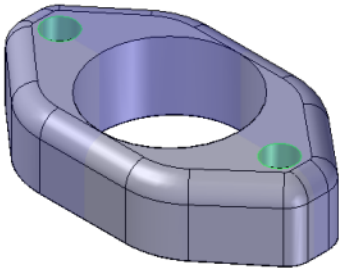


Figure 3.25: Final EEG Holder Design for New System

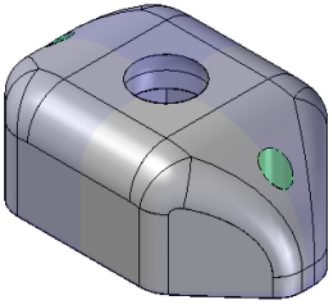


Figure 3.26: Final NIRS and Ball Lens Holder for New System

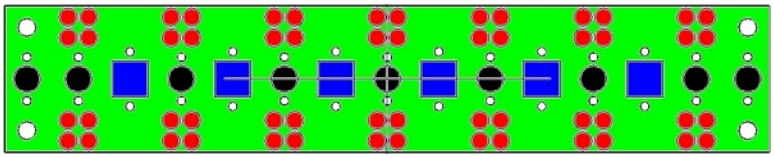


Figure 3.27: Final Layout for 3rd Generation Sensor Board (Red: LEDs, Blue: Photodiode, Black: EEG)

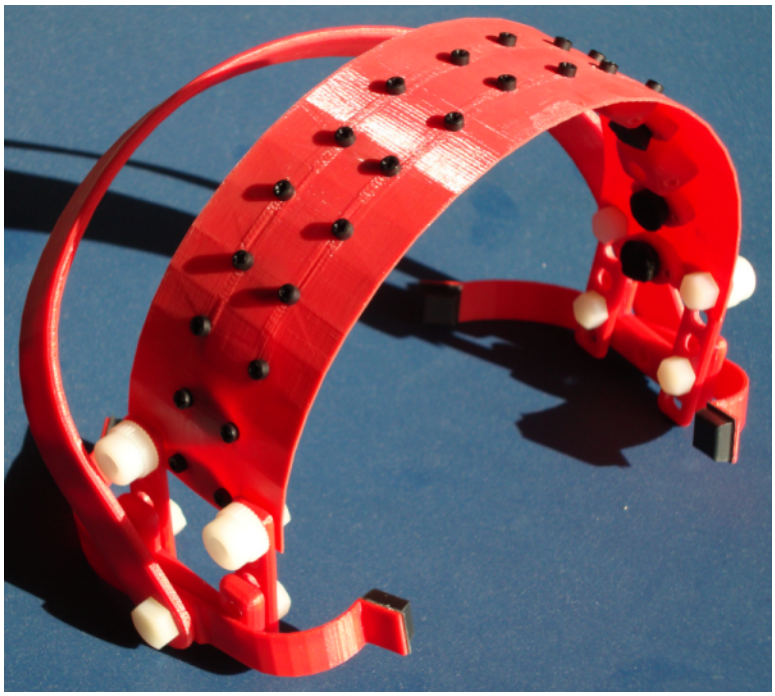


Figure 3.28: Completed Structure for Third Generation Headset with Dummy Circuit Board Insert

CHAPTER FOUR: CONTROL DESIGN

Preliminary Data Analysis

In order to make certain that a simple relationship between brain activity and limb function does not exist, a series of training sessions were performed under different sensor conditions. The combined data was plotted for each input channel, correlating the intensity of the FFT response to each separate output activity. Four input EEG channels were used and FFT frequencies 1-15 Hz were mapped individually to each of the four EMG output channels. Additionally, box plots were prepared, indicating the level of variation observed on each input frequency, classified by the output activity. Consolidated graphs representing this data are available in Figures 4.1-9. The dip that occurs in the FFT data at specific EMG intensities is the result of splitting the data file based on output activity for the box plots.

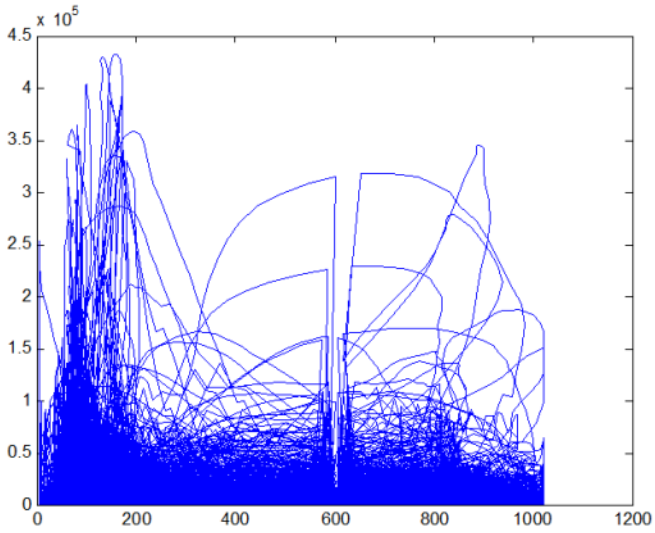


Figure 4.1: FFT Indices 1-60 vs. EMG Channel 1

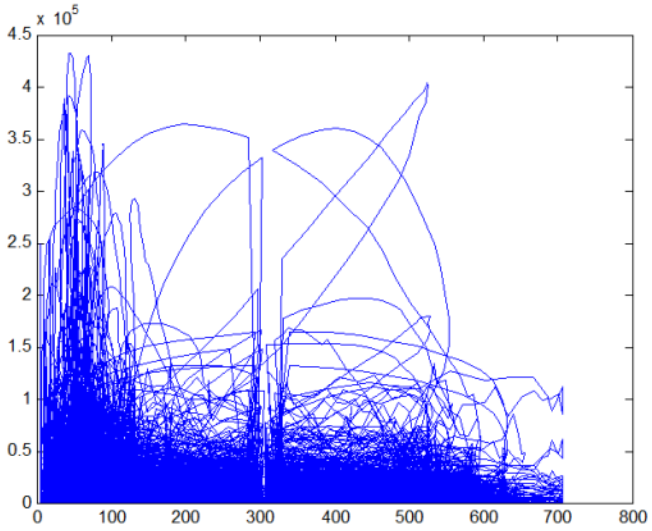


Figure 4.2: FFT Indices 1-60 vs. EMG Channel 2

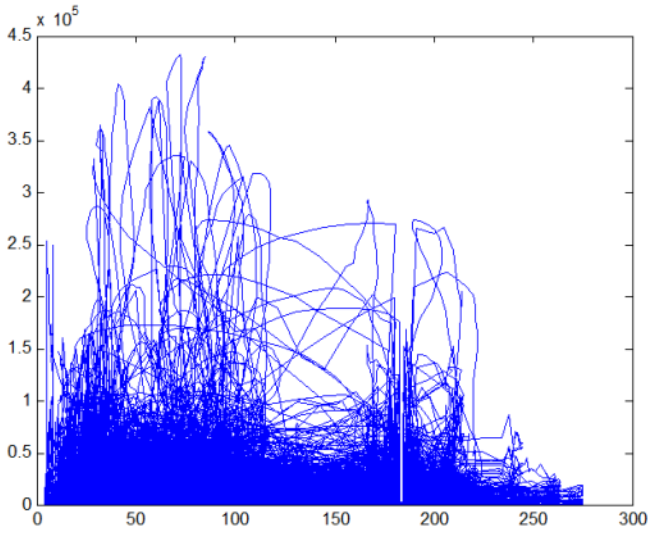


Figure 4.3: FFT Indices 1-60 vs. EMG Channel 3

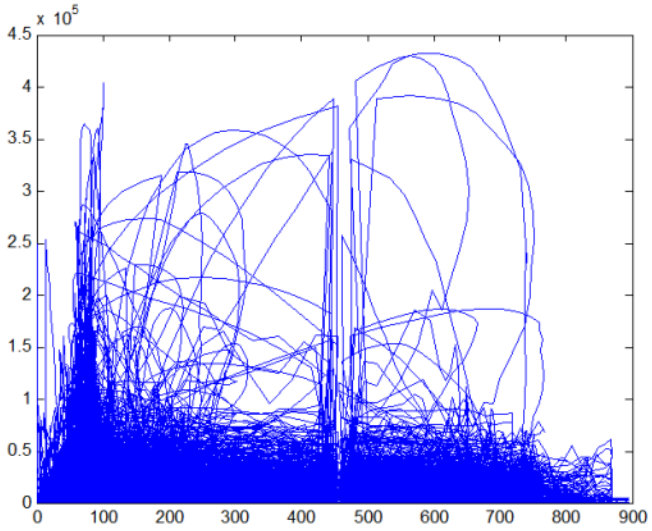


Figure 4.4: FFT Indices 1-60 vs. EMG Channel 4

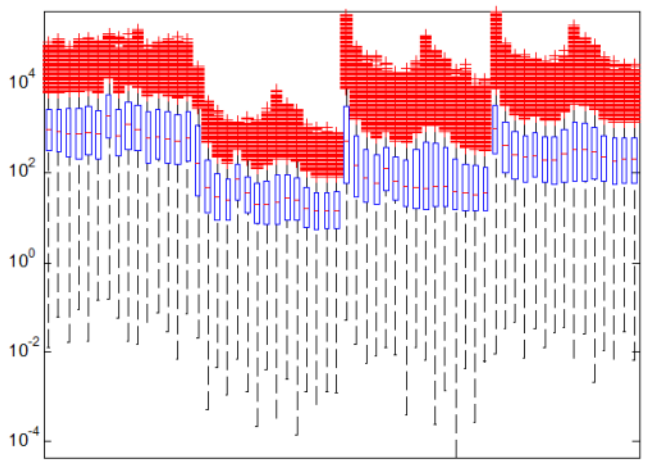


Figure 4.5: FFT Indices 1-60, Intensity Between Activities

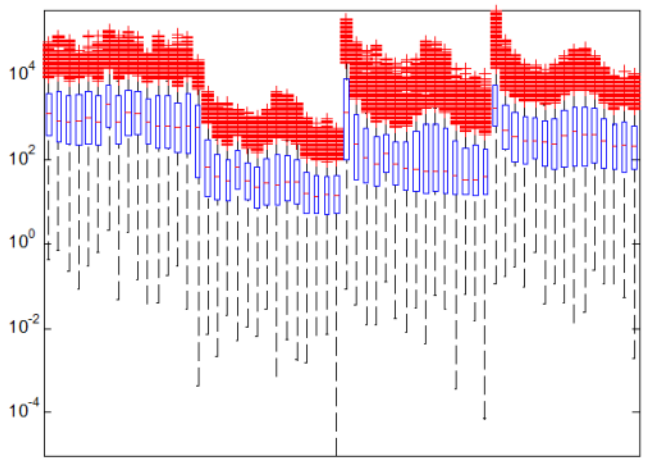


Figure 4.6: FFT Indices 1-60, Intensity During Bicep Flexion

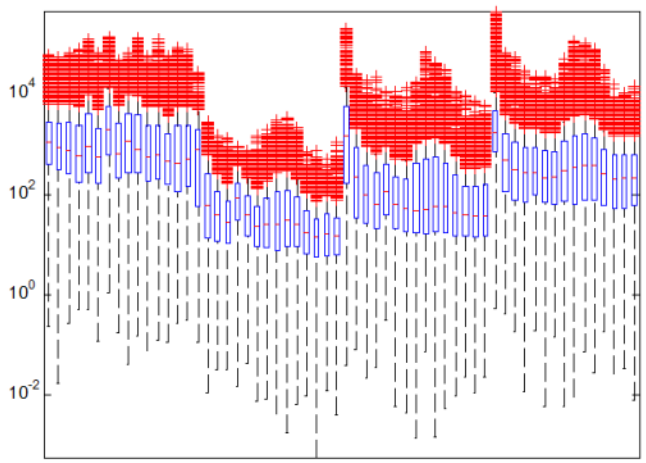


Figure 4.7: FFT Indices 1-60, Intensity During Triceps Extension

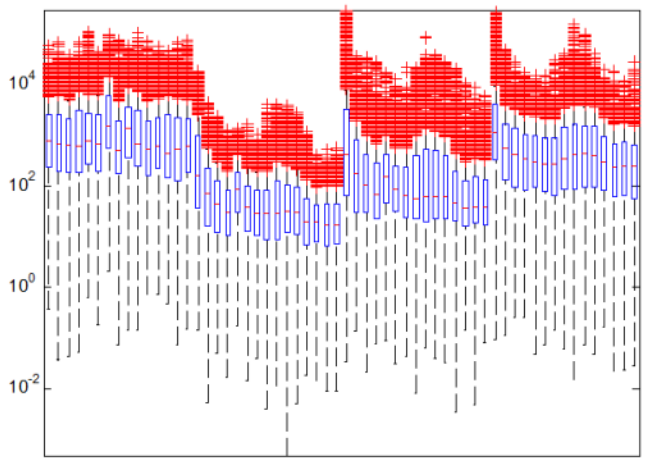


Figure 4.8: FFT Indices 1-60, Intensity During Wrist Flexion

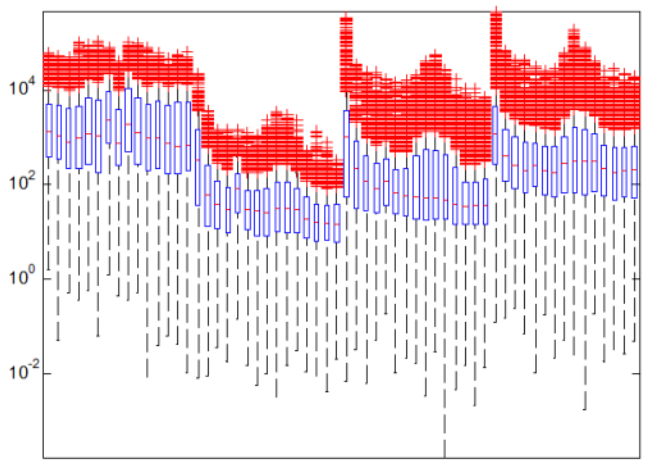


Figure 4.9: FFT Indices 1-60, Intensity During Wrist Extension

From the uncondensed 240 input-to-output plots and 60 box plots it can be determined that no clear, simple relationship exists between any of the inputs and outputs. If any relationship appears in the data, it may be an inverse correlation, but lacks a distinct slope. For this reason, investigating a more sophisticated learning algorithm to perform linear and non-linear mapping of input to output data is justified.

Transfer Function

To first determine if a linear regression model was sufficient for mapping the inputs and outputs of this system, a transfer function was designed and trained using Matlab. The transfer function trains weighting constants that relate each frequency component of the system inputs to the intensity of the simultaneous EMG recordings. Figure 4.10 shows the system inputs and outputs of this control algorithm.

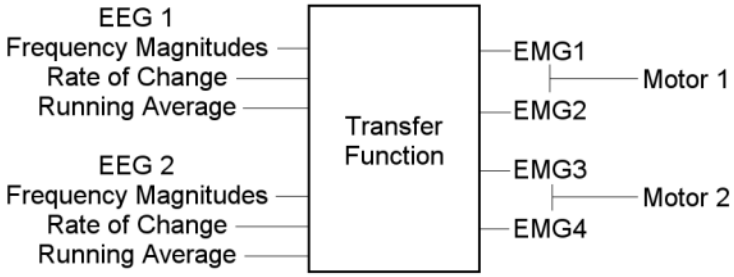


Figure 4.10: Transfer Function System Diagram

The transfer function determines the intended muscle activation based on real-time data being collected via EEG. The EEG is analyzed to determine the frequency composition, rate of change, and running average of the signal and those features are used along with the coefficients produced by the trained transfer function to determine the output values for EMG channels one through four by multiplying the product of the FFT components with the variance and running average contributions. The EMG outputs are then combined to produce the desired rotation of their corresponding joints, which is used to provide the control signal for the motors. This method produces scaling values for motor control that correlate with strength of relevant brain activity.

The tests using a transfer function to calculate desired muscle activity from EEG recordings indicate that this method is suitable for controlling the artificial limb during very simple motions. The best results were obtained by training the transfer function using a variety of tasks to create the most diverse mapping of brain activity to muscle activation and then using the trained constants to calculate motor outputs for the elbow and wrist joints during simple tasks. The EEG frequency response to EMG signal strength correlations for the control system used in this set of results is presented in Figure 4.11 and 4.12. These figures demonstrate the significance of certain frequency

ranges on determining which muscle group is currently being actuated. While these results have a similar distribution to the EEG results presented from the human subject study, the resolution of this analysis is more decisive than the original method.

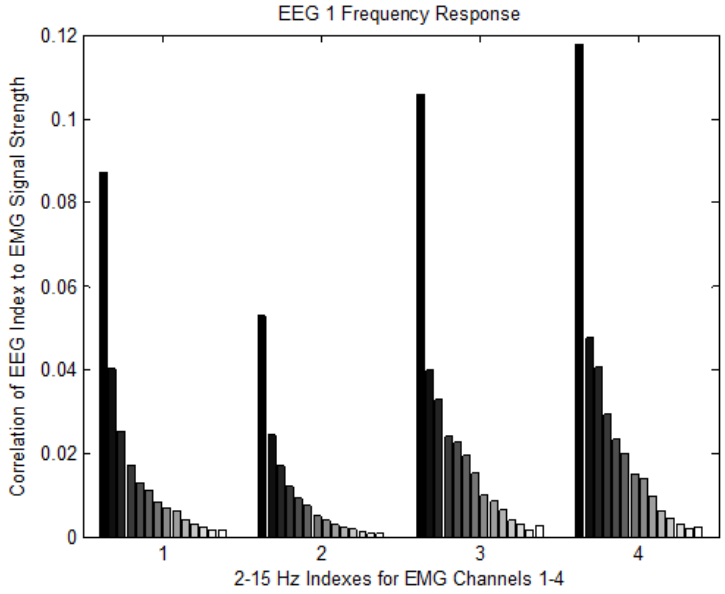


Figure 4.11: Transfer Function Index Weightings Correlating EEG 1 Frequency Composition to Signal Strength at EMG Channels

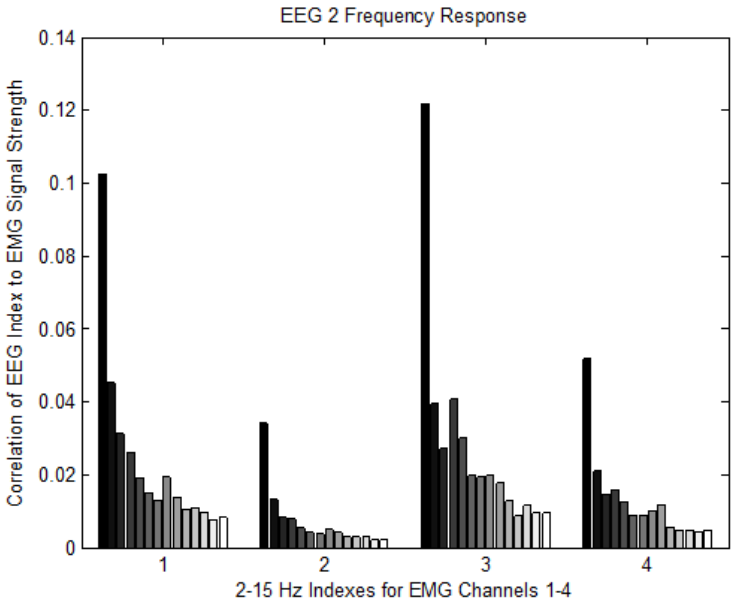


Figure 4.12: Transfer Function Index Weightings Correlating EEG 2 Frequency Composition to Signal Strength at EMG Channels

The signal magnitude and variance (Figures 4.13 and 4.14) play a smaller role in determining which muscle groups are currently being used, but play a larger role in determining how strong of an actuation is occurring. These contribute to attenuating the effect that larger muscle groups have on EMG signal strength as well as compensating for the relative strength of movement a subject uses during transfer function training. In the figure below, it is apparent that a stronger correlation between bicep actuation (EMG 1) and signal magnitude was observed compared to triceps actuation (EMG 2).

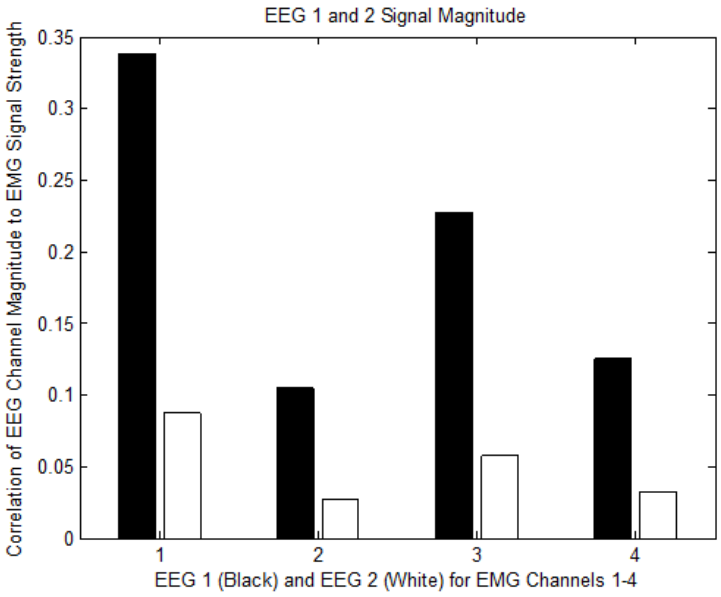


Figure 4.13: Transfer Function Index Weightings Correlating EEG 1 and 2 Signal Magnitude to Signal Strength at EMG Channels

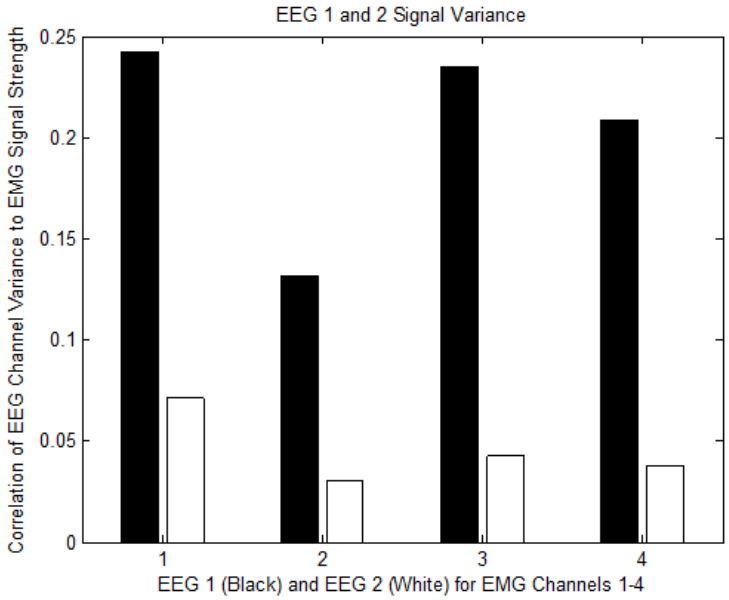


Figure 4.14: Transfer Function Index Weightings Correlating EEG 1 and 2 Signal Variance to Signal Strength at EMG Channels

While sufficient output matching was acquired during very simple movements using the transfer function method, there was very poor distinction between output activities, resulting in bicep flexion during most wrist movements. To improve the performance of the simple control system, the following changes were made to attempt to distinguish between activities more consistently.

The actuators in the test arm being constructed (discussed in chapter 5) are two servo-motors that are position controlled with pulse width modulation (data sheet available in Appendix A.7). The digital to analog converter that controls this signal works on a scale ranging from 0 to 1023, where 512 is the neutral position for both joints, so the system output was designed to match these ranges. The output value for each motor is calculated using equation 4.1. A simple proportional error controller was used for the test arm to limit the rate of actuation.

$$P = P + G * (D - P) \tag{4.1}$$

Where P is the motor position, G is a calculated gain based on the sampling rate, and D is the desired position for each motor. If P is outside of the bounds of the minimum and maximum set for that joint, P is set to the corresponding bound.

The desired position D is set by equation 4.2 and is the calculated estimate of the EMG values that were used to train the system.

$$D = E_{k1} * E_1 - E_{k2} * E_2 \tag{4.2}$$

Where E_{k1} and E_{k2} are selected scalar constants based on EMG signal intensity and E_1 and E_2 are the calculated EMG values for opposing sets of muscles corresponding with the selected joint. This functions the same as the human body, where opposing muscle groups produce moments about a joint and the differential of the moments determines the force of the output (Winter 2005, 87). The calculated EMG values are determined by equation 4.3.

$$E_n = E_t * \frac{\sum_m F_{km} * F_m}{F_t} * \frac{V_k * V}{V_t} * \frac{S_k * S}{S_t} \tag{4.3}$$

Where F, V, and S represent the FFT, variance, and summation of the EEG input data. The variance is the difference of the maximum and minimum values observed on an input channel during the last 128 samples and the summation is the first index of the FFT for each input channel. E_t, F_t, V_t, and S_t are threshold values for corresponding terms based on maximum values observed during training. F_k, V_k, and S_k are matrices correlating the relationship of each EEG input measure to corresponding EMG output values. The subscript n indicates the EMG channel that the calculation is being

performed for and the subscript m indicates the EEG channel. These matrices are normalized prior to use.

The transfer function is trained by building a correlation between measured inputs and known outputs using the following algorithm. Equations 4.4 – 4.6 are used to train F_k, V_k, and S_k.

$$F_k = F_k + T_k * \frac{E - E_m}{E_t} * \frac{F}{F_t} \tag{4.4}$$

$$V_k = V_k + T_k * \frac{E - E_m}{E_t} * \frac{V}{V_t} \tag{4.5}$$

$$S_k = S_k + T_k * \frac{E - E_m}{E_t} * \frac{S}{S_t} \tag{4.6}$$

Where T_k is a training constant based on the number of data points in the training set, E is the measured EMG, and E_m is a matrix of minimum values observed on each EMG channel.

The best results are obtained by training the transfer function using a variety of tasks to create the most diverse mapping of brain activity to muscle activation. The trained constants are then used to calculate motor outputs for the elbow and wrist joints during simple tasks. The system was tested using healthy subject data that had previously been collected. Measured EMG activity during brain monitoring sessions was used to validate the output of the EEG to EMG transfer function.

To minimize the effect that changes in sensor connection quality have on the transfer function compatibility, the sensor system is recalibrated each time it is used,

allowing for the change in signal strength to be the dominating factor and not absolute magnitude. This led to the development of an automated system for establishing system parameters.

Two major limitations of the original transfer function training system were the requirement of the system to be trained for each new individual and for values to be selected for the relative effect that each component of the system should have on the output. The combination of two baseline recordings is used to overcome these limitations. A resting baseline paired with a maximum activation measurement allows for the automated selection of relative weightings as well as the selection of a potential matching trained system selected from a database.

The threshold, minimum, and maximum values that were previously manually set are selected by characterizing the signal properties automatically. As a result most of the constants that were experimentally determined in the original control system are now automatically calculated at each session based on the baselines, using the components of the data that were originally used to derive those values.

The effect that each calculated EMG value has on the net motor output is also adjusted during the training session. The training activities are assumed to produce maximum flexion of each joint in both directions, therefore the relationship between opposing EMG outputs is modified to achieve a maximum flexion based on the values acquired during the training activity. The value of each EMG channel is recorded at the peak of each activity and stored in the matrix E_t. The values for E_k are then calculated using equations 4.7 – 4.10 to simultaneously solve for the variable sets.

$$E_{k2} = \frac{M_{t1,2} * E_{t1,1} - M_{t1,1} * E_{t2,1}}{E_{t1,2} * E_{t2,1} - E_{t2,2} * E_{t1,1}} \tag{4.7}$$

$$E_{k1} = \frac{M_{t1,1} + E_{k2} * E_{t1,2}}{E_{t1,1}} \tag{4.8}$$

$$E_{k4} = \frac{M_{t2,2} * E_{t3,3} - M_{t2,1} * E_{t4,3}}{E_{t3,4} * E_{t4,3} - E_{t4,4} * E_{t3,3}} \tag{4.9}$$

$$E_{k3} = \frac{M_{t2,1} + E_{k4} * E_{t3,4}}{E_{t3,3}} \tag{4.10}$$

Where M_t contains the maximum EMG values recorded for each channel.

A second element has also been added to the training algorithms to improve the correlation between input signal strength and net output. Equations 4.11 and 4.12 are used to adjust the training matrix values based on the observed error between calculated EMG output and measured EMG for each channel. This adjustment has led to the elimination of using the EEG Summation term in the EMG output calculations since the main purpose of that term was to monitor the relative correlation of the intensity of each input to output.

$$F_{kn} = F_{kn} + T_k * (K_n - E_n) \tag{4.11}$$

$$V_{kn} = V_{kn} + T_k * (K_n - E_n) \tag{4.12}$$

Where K_n is the known EMG output and the n^{th} set of elements is modified to better correspond with the n^{th} output.

If the user does not have the ability to attach EMG sensors to a limb, the training exercise is performed with visual stimulus while the user tries to mimic the

corresponding output. The artificial limb performs a pre-established set of motions corresponding to the same activities that a healthy subject would have performed during the training session.

NIRS is used during real-time operation to further adjust the output of the system to better match the activities and desires of the user. The coefficients of the proportional error controller are modified continuously based on the relative brain activity measured through NIRS by making adjustments if there is deviation in relative brain activity compared to motor output. If the NIRS data matches a sustained muscle activity and the recent EEG data isn't producing the same output, the motor transfer function values are adjusted incrementally to adapt the system to better match the desired system output within a bound of values allowing up to two orders of magnitude variation.

Further research has lead to the decision to implement a different technique for analyzing the input signals from EEG. Instead of observing each input channel and correlating the characteristics to output data, combining paired sets of sensors into groups and analyzing the difference between channels as well as their average has shown improved results (Collura, 2005). The new control system inputs are obtained with equations 4.13 – 4.16.

$$F_1' = 512 + (F_1 - F_2) \tag{4.13}$$

$$F'_2 = \frac{1}{2}(I_1 + I_2) \tag{4.14}$$

$$F_3' = 512 + (F_3 - F_4) \tag{4.15}$$

$$F_4' = \frac{1}{2}(F_3 + F_4) \tag{4.16}$$

Where F_n' represents the new input used for the control system and F_n represents the original FFT input data from each EEG channel. The values 512 and 1/2 are used to re-center the data on the 0 to 1023 input data range being used.

Case Matching Algorithm

The case matching method uses a training algorithm to learn the expected frequency composition, rate of change, and running average of the two EEG signals for each motor activation case using Matlab. The simplest implementation of this method uses five cases that correspond to large motions. These cases correspond to elbow flexion, elbow extension, wrist flexion, wrist extension, and resting. After training the case variables using trial data, the control algorithm compares the current frequency composition, rate of change, and running average of the two EEG signals to the case variables and determines which case has the least absolute error. If the least absolute error is greater than a threshold the decision is “no match.” If one of the cases is a close match to current signal parameters, then the desired position of the motors corresponding to the case’s training is set to the case’s actuation value (Figure 4.15). By adding many cases to this determination, complex motor activations can be achieved. However, as the number of cases being compared to the current state of the signal increases, the processing power required increases and the use of a general nearest neighbors algorithm would be more effective.

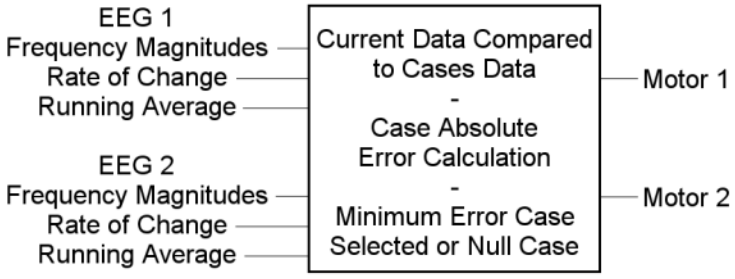


Figure 4.15: Case Selection System Diagram

Control System

Based on the results of the preliminary studies, it was determined that a control system using EEG as the primary input should be sufficient for real-time control of artificial limbs. EMG is used to produce known output data for learning as well as performance analysis. NIRS is used as a feedback element for real time adaptation in the event that EMG is not available. The flow of the inputs and outputs to and from the control system is available in Figure 4.16, where the control system block represents any of the learning algorithm investigated in this study.

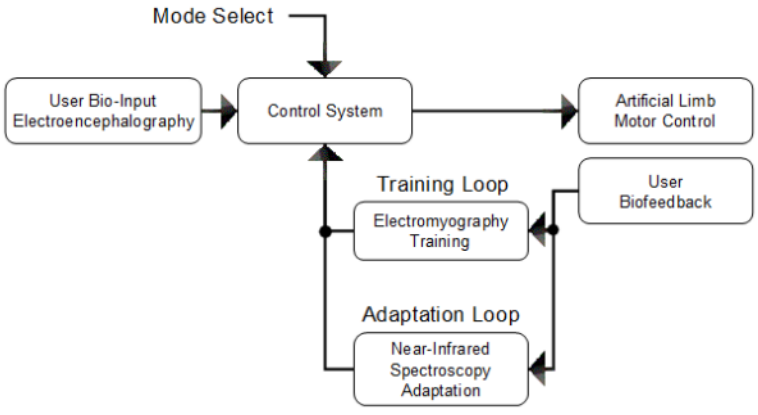


Figure 4.16: General Control System Sensor Flow Diagram

CHAPTER FIVE: LEARNING AND ADAPTATION

Learning

Learning is necessary for this application because there is too much variation in the biological signals collected from healthy subjects in the preliminary studies. Furthermore, there is no clear distinction between input and output relationships to establish a generalized system form. For these reasons, a machine learning algorithm is most appropriate for developing a control system in this study. A fuzzy-nearest neighbors algorithm, a k-nearest neighbors program, and both linear and non-linear neural networks will be evaluated. Learning is performed using the FFT of EEG data or the PCA of the FFT data as the system inputs and EMG data as the known outputs, with a simple difference of magnitude used to determine the joint actuation based on EMG activity.

Fuzzy-Nearest Neighbors

After investigating the simple control methods, a slightly more complicated linear system was created based on the results from the earlier studies. The result is a combination of various techniques that achieve certain goals for our objective and aims to produce a nearest neighbors algorithm that relies on generalization of member neighbors, which should reduce run-time computational demands by performing some learning during the training process. The structure of the control process may look similar to a neural network (Figure 5.1), but the functionality is closer to a nearest neighbor

algorithm. We use a summation of all training data points that are similar enough to the current readings to be considered possible matches, producing a high-pass filter effect for the contributions of the training set data to the estimated output. The combined output estimation is then analyzed and the most likely desired output is selected and scaled for output to the motor control. The details of this process are described in the following sections.

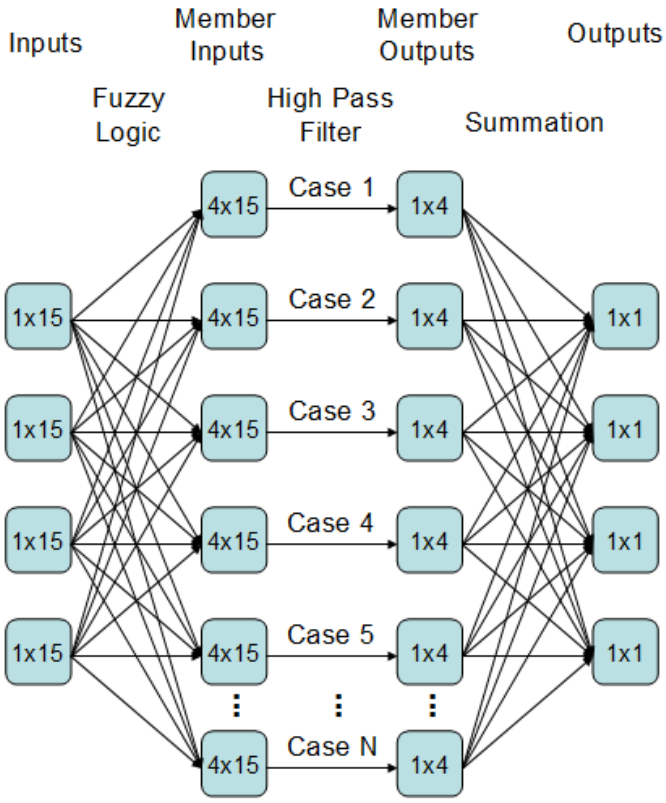


Figure 5.1: Fuzzy Neighbors Control System Diagram

At the initiation of real-time control, certain variables are prepared, notably the current EEG data array and the current motor values, which need to be set back to zero to avoid complications with initial analysis. The headset is turned on and based on the response from the serial routine, an error message is displayed or data processing begins. The number of samples that are sent to the computer from the headset is determined

based on the length of the data stream being received and the number of sample sets is returned to the main control routine to keep track of the current time index. The magnitude of the FFT indices for one through fifteen Hertz are obtained and stored in the input layer matrix.

Based on the comparison of the current magnitude of each index with the maximum magnitude for that index during baseline and the first logic determination value, the index is converted into a fuzzy logic value and stored in the input layer logic matrix. The fuzzy values [0, 0.25, 0.5, 0.75, 1] are assigned on the criteria [>0, >0.1, >1, >10, >100] times the product of the maximum magnitude and the first logic determination value. At this point the current sensor data preparation has been completed and a call to a sub routine to perform the control calculations is made.

The input layer logic matrix is compared to every training matrix and one minus the difference between logic indices is summed and stored in the input layer matrix (Equation 5.1). If any input layer matrix exceeds the cut-off for being a possible activity match, then the value is manipulated to represent the certainty of the match and stored in the input layer matrix and as a one in the input layer logic matrix. If the value does not exceed the cut-off, the value in both matrices is set to zero. The certainty of the match is determined by Equation 5.2.

$$FL_i = \sum 1 - (L_j - T_{ij}) \tag{5.1}$$

Where FL_i is the first layer summation value, L_j is the logic value of index j, and T_{ij} is the corresponding training matrix indices.

$$I = \frac{FL_i \big/ FL_{\max} - LD_2}{1 - LD_2} \tag{5.2}$$

Where $FL_{\max}$ is the maximum possible value, and LD_2 is the second logic determinate, which is used for the cut-off.

The reason two values are created for each value of the first layer is because at this point there are two modes of control that can be used. The first is a match confidence ranging from zero to one, while the second is a pure logic simplification. Based on the mode being used, the output layer values are either accumulated using the first layer or the first layer logic matrices, then multiplied by the pre-trained second layer matrix.

The output is then passed through a winner takes all routine to determine the most significant output estimation. All other values are set to zero. Each channel is then multiplied by a gain factor based on the number of active nodes, and if the value does not exceed a minimum value, it is dismissed as noise and all output values will be zero.

The next step is to scale the output value based on the available range of motion for that limb segment (Equations 5.3).

$$O_i = O_i * \frac{(J_{mid} - J_{\max})}{1023} \tag{5.3}$$

Where O_i indicates the output value for the EMG channel, J_{mid} is the neutral position for that joint, and $J_{\max}$ is the maximum rotation for that output. The mid and max are intentionally arranged to produce a negative value for a positive flexion due to the orientation of the motors.

Then the error between the current motor position and the output of the control algorithm is determined (Equation 5.4) and input to a proportional-integral-derivative (PID) controller (Equation 5.5) for all four motors. Motor error is found by the difference of the current position and the net effect that the output channels have on that limb segment. For example, motor zero represents the position of the elbow, which is a function of the difference of the bicep and triceps, which are represented by output activities zero and one.

$$E_i = (O_i - O_{i+1}) - M_i \tag{5.4}$$

Where Ei represents the motor output error, O indicates the EMG output, and M represents the current motor position.

$$M_i = k_p * E_i - k_d * v_i + k_i * \sum_0^t E_i \tag{5.5}$$

Where M_i represents the new motor position, k_p is the proportional control constant, k_d is the derivative control constant, v_i is the rate of change for the motor, k_i is the integral control constant, and $\sum_0^t E_i$ is the error summation on the motor channel.

For the purpose of diagnostics, the value being sent to each motor, the current sampling time, and a measure of the EEG activity are printed to the screen. If adaptation is turned on, it is at this time that the sub routine for comparing NIRS information with the current system output is performed.

The training system built into the fuzzy neighbors program has become one of the main methods of gathering sample data for this project. When this function is called, the user must first select if they will be training with or without EMG, which was added for

the purpose of collecting data using visual feedback when an EMG signal can not be collected for any reason, primarily in the case of limb loss. The program then records the current time and uses it to create a new file, which will point to subsequent files created during the training session. Once the training menu has loaded, the options to train bicep flexion, triceps extension, wrist flexion, wrist extension, forearm torsion, and hand grasping are available as well as returning to the main menu. When one of the training tasks has been initiated, the program then calls to a subroutine, which receives an integer corresponding to the activity to be performed as well as the mode of operation. When the training menu is exited, the file pointing to the sub-training files is closed and some of the relevant baseline recording variables are stored to the same file for use when the training session is called for later use.

The task training function also records the current system time and creates the task-specific file containing the data points used for training during the session. The headset is then activated and data is recorded from EEG, NIRS, and EMG until a sufficient number of points have been collected to fully train the fuzzy-neighbors control system. As the training is performed, the training progress is printed to screen to let the user or overseer know if the correct intensity of activity is being performed. These intensity thresholds are set in the .ini file that the program opens at each initiation. If the system is operating without EMG, the motors are fed an oscillating activity corresponding to the training being performed and the user is required to attempt to move their limb in the same pattern as the visual stimulus. This way the motor output values are substituted for the EMG data that would normally be used during a training session. At

the end of the task training, a report is generated reflecting the current progress of training the entire fuzzy-neighbors algorithm. If an activity is repeated it will overwrite the data in the variable space for that activity, but two separate output files will be recorded. Once the task is complete, the user is returned to the training sub menu.

Once a training session has been completed, the accuracy of the system can be improved by running the function to train from file. This option prompts the user to input a training session file to use for optimization as well as one to be used for testing. This will then pull all six task training records from the training session file as well as import the baseline variables that were in-use at the time of the recording. A record file is also generated based on the current time, which will record all of the data produced by the optimization process at the time the menu is exited.

This sub menu has two options which are "Fast Solution" and "Test Solution." These are the ways to optimize the algorithms performance and evaluate its effectiveness that are currently being used. Fast solution circumvents refining the first determination value and simply sets the target to .50, which is a generally favorable value, particularly when using fuzzy-logic instead of simple logic, input values. The program then calls the function to find the logic determination value that will produce the average of the fuzzy input values to be 0.50. This will provide an even spread of fuzzy input values to maximize the algorithms ability to determine match quality.

The next function to be called pulls the data from the task training files and uses it to populate the fuzzy neighbors known cases with logic simplification based on the logic determination value found in the previous step. Now that known input values are

available, the value of the second logic determination cutoff is optimized by recursively seeking out the logic determination values that produce the highest accuracy rate when testing against the training set. After the process is complete, the runtime is printed to the screen along with a report of the determination value selected and its performance.

During each step of the refinement processes, the accuracy of the system is saved to a variable matrix along with the parameters used in that pass. These matrices are written to the record file created when entering the training from file sub menu and are used to analyze the behavior of the fuzzy neighbors determination values by plotting how accuracy trends (Figure 5.2).

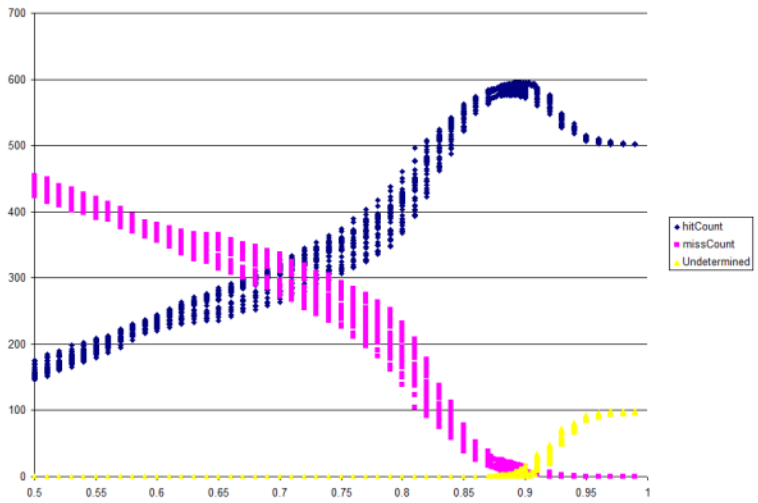


Figure 5.2: Plot of Fuzzy Neighbors Self-Test Accuracy vs. Second Determination Value

When the test solution function is called, the same process is used to pull data from the task training sessions that are now being used to test the system. The test samples are turned into fuzzy values based on the criteria from the training from file session and the output of the fuzzy-neighbors control system is compared to the value in the test file. The number of hits, misses, and undetermined values is recorded along with

the average absolute error for hits and misses, which are printed to screen. This algorithm uses a generalized activity power calculation to determine the signal strength, because it allows the system to perform forearm torsion and hand grasping determinations, which don't have a clear EMG value in this data acquisition system. The activity power is simply the summation of the EMG sensor values during a known action.

k-Nearest Neighbor

To evaluate the effectiveness of an existing, simple machine learning technique as the control element for our limb actuation system, a basic k-nearest neighbor program was written in C++ to operate on several sample sets of data using EEG and EMG as input and outputs for the system (Figure 5.3). k-Nearest Neighbor was chosen for its very simple design as a lazy-learning method, to examine how well separated the input data was based on output classification. The design of the program is detailed in this section.

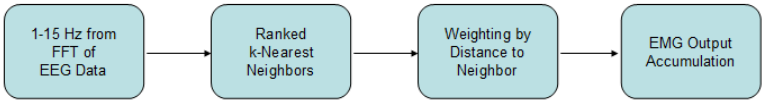


Figure 5.3: k-Nearest Neighbor Control Flow Diagram

Upon starting the program, an initialization function runs immediately before the menu appears. The sample vector is used to establish a list of indices corresponding to the length of the data file being used. Then random indices of the sample vector are selected and inserted into a test vector. When an index has been used, the value at that index is set out of range and recognized to be excluded from future random pulls from the sample vector. Indices that are not selected to be in the test vector are then used for the training vector. This is a misnomer since k-nearest neighbor does not actually train on the data

set, but these values are used for the ‘known cases’ that the k-nearest neighbor algorithm knows both the input and output for. The data file is now opened and the data points corresponding to testing and training sets are separated into their variable matrices.

After the data has been loaded into the variable space, the menu loads and prompts the user to exit or start the nearest neighbor program, which simply tests the control algorithm, since no training is required. The program runs the evaluation of every test data set using the following methods.

The distance between the test point and every point in the known data set is calculated in Cartesian space by summing the square of the difference between all sixty input channels (Equation 5.6).

$$D_n = \sqrt{\Sigma(F_i - FN_i)^2} \tag{5.6}$$

Where D_n is the distance to neighbor n, F_i is the sample FFT magnitude on each index and FN_i is the corresponding neighbor FFT magnitude.

Once all of the distances have been accumulated, the k-nearest neighbors are determined by seeking out the smallest value in the distance array and then removing that value from the list of candidates. Because of this step, there are two distance vectors in memory, so one remains untouched. The test point is then classified by combining the weighted contribution of the k-nearest neighbors (Equation 5.7). The weightings are determined by dividing one by the distance to the neighbor.

$$C_i = \Sigma(\frac{1}{D_n}) \tag{5.7}$$

Where C_i is the classification array used to accumulate the votes of the k-nearest neighbors.

The classification with the highest vote count is then assigned as the class of the test point and if the known class of the test point is the same as the k-nearest neighbor classification, the accuracy tracking variable is incremented.

The next step in the program is to attempt to perform regression based on the values in the known set. The output values corresponding to the k-nearest neighbors are averaged without any weighting (Equation 5.8) and the error between the actual output of the test point and the calculated value are compared.

$$E_i = \Sigma \frac{EN_i}{k} \tag{5.8}$$

The square of this error is accumulated across the test set to determine the root mean square error (RMSE). Once all of the test points have been evaluated, the program prints to the console interface the RMSE for each output channel as well as the accuracy of this test set.

Linear Activation Neural Network

A linear neural network was examined prior to more conventional neural network designs, because it would reduce the computational requirements of the system by removing the need to run every node through an activation function as well as reducing the formatting of the input and output data for training and implementation. Additionally, by using a linear network, the output layer values could be used immediately for control,

without any linearization of the output values. A diagram of this neural network is available in Figure 5.4.

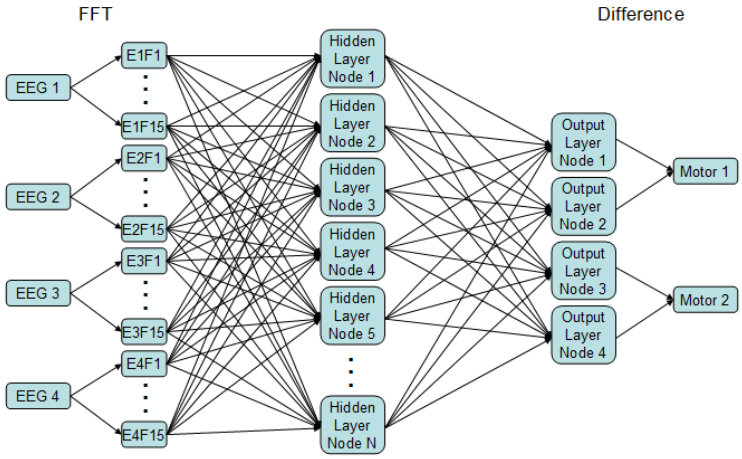


Figure 5.4: Linear Actuation Neural Network Control System Diagram

This program was written in C++ to facilitate use with existing code. The program begins by loading the most recent neural network weightings file each time the program is initiated and saves the current weights to file each time the program is properly exited. As long as the number of hidden nodes has not changed since the last training session, the neural network will already be functional at initiation. The user console provides a menu with the options to load a weightings file or train from file. Loading weightings data allows the user to type in the name of previously recorded neural networks and the initial loading operation will take place using the new data.

Training from file originally prompted the user for train and test file locations, but repeated loading became tedious, so the file names are currently written into the code and are loaded automatically. The train from file menu provides the options to randomize the network weightings, train the network, and test the network. Randomizing the weights generates random numbers between -1 and 1 in 0.001 increments. This type of random

distribution is referred to as having a “hard” range of values and may not be as optimal as a Gaussian distribution or several other techniques that are used for neural network initialization, but was sufficient for our testing (de Castro 2001).

Selecting the network training option runs the learning algorithm for the neural network. Data is read from file each training cycle to mimic data acquisition from the headset. After variables used to accumulate values have been zeroed, the forward propagation process begins using equations 5.9 and 5.10.

$$H_i = \Sigma(F_{jk} * wh_{ijk}) \tag{5.9}$$

Where H_i are the hidden layer nodes, F_{jk} are the FFT inputs, and wh_{ijk} are the weightings from the input layer to the hidden layer.

$$O_i = \Sigma(H_i * wo_{ij}) \tag{5.10}$$

Where O_i are the output layer neurons and wo_{ij} are the weightings from the hidden layer to the output layer.

The next step is error calculation and back propagation using equations 5.11 – 5.13 and the total mean absolute error is printed to screen and the value is returned to the parent function which will run until the error reaches a tolerance.

$$wo_{ij} = wo_{ij} + LR * (T_j - O_j) * H_i \tag{5.11}$$

Where LR is the learning rate and T_j is the teacher output value.

$$\Delta H_i = \sum (T_j - O_j) * wo_{ij} \tag{5.12}$$

Where ΔH_i is the back propogated error at the hidden nodes.

$$wh_{ijk} = wh_{ijk} + LR * \Delta H_i * F_{jk} \tag{5.13}$$

Testing the solution pulls data from the test file instead of the training file, but functionally performs the same forward propagation, partial back propagation, and error calculations, but without modifying the weighting values. This allows the user to test the neural network against a completely unique, but similar set of data.

Principle Component Analysis

This program only performs one function, performing a PCA on a dataset. It was written in C++ to evaluate the usefulness of PCA on this data type and to verify the methods used prior to insertion into a neural network program. At run time the program generates a new file name every time the program is run so that a unique output file is created. Once the main menu has loaded, the only options are saving and exiting the program, which will save any PCA data to file, or to run a principle component analysis. When the function to generate PCA values is run, the program loads a file name that is hard coded into the program. The option to type in a file name each time is currently removed, but can be included later to add functionality. Since training data is being used in this application, the same methods of pulling task, FFT, EMG, and NIRS data from recorded file are used. Once the data is loaded, the PCA subroutine function begins and the actual PCA is performed.

First the mean value of each input channel is calculated (Equation 5.14), then the mean is subtracted so that the input data is mean-centered (Equation 5.15).

$$M_j = \sum_i \frac{F_{ij}}{S} \tag{5.14}$$

Where M_j is the mean for an input channel, F_{ij} is the FFT input values, and S is the nunber of data sets being used.

$$C_{ij} = F_{ij} - M_j \tag{5.15}$$

Where C_{ij} are the centered values for each data point.

Once this is complete the covariance matrix can be calculated (Equation 5.16).

$$Cov_{ij} = \sum_k \frac{C_{ki} * C_{kj}}{S-1} \tag{5.16}$$

Where Cov_{ij} is the covariance between input channels i and j.

Next the covariance matrix is imported into an Eigen-type matrix so that the included files to perform Eigen analysis can be used (Eigen is LGPL-licensed). Now that the data has been prepared, EigenSolver can be used and Eigenvalue and Eigenvector matrices are generated. An easy way to extract these values from the Eigen data type was not found, so an output file stream was used to write the values to file. Now that the data exists in a file, simple knowledge of the formatting allows the data to be pulled back from file and stored in more convenient data structures.

After the Eigenvalues and Eigenvectors have been acquired, the next step is to sort the Eigenvalues by magnitude. The Eigenvectors can then be sorted using the corresponding sorted Eigenvalues and organized into the Feature Vector (Equation 5.17).

$$FV_{ij} = EV_{iS(j)} \tag{5.17}$$

Where FV is the feature vector, EV is the Eigenvector matrix, and $S(j)$ is the sort array from the Eigenvalue sort process.

Calculating the principle component values of the data is now only a matter of matrix multiplication (Equation 5.18).

$$P^T = FV^T * F^T \tag{5.18}$$

Where P^T are the new principle component values, transposed, FV^T is the transpose of the feature vector, and F^T are the transpose of the FFT input values.

Sigmoid Activation Neural Network

Using a sigmoid activation function requires more computation, but adds a non-linear element to the control process, which can find a solution to non-linear problems. Based on the results of our initial analysis of the relationship between the inputs and outputs of this system, it could be assumed that a non-linear solution such as this one would be required, but more simple solutions needed to be tested as well to avoid over-complicating the control system. The non-linear neural network is designed to work both without (Figure 5.5) and with principle component analysis of the input values (Figure 5.6). PCA is used here to reduce the dimensionality of the input array as well as rotate the data into the coordinate system that produces the greatest variation.

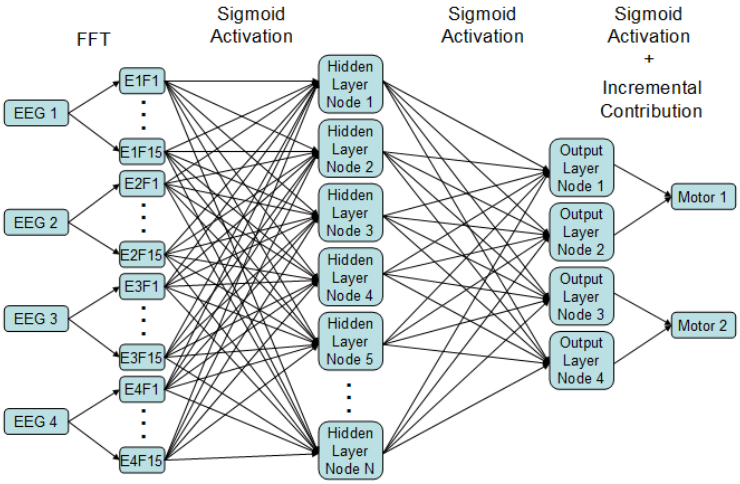


Figure 5.5: Sigmoid Activation Neural Network Control System Diagram

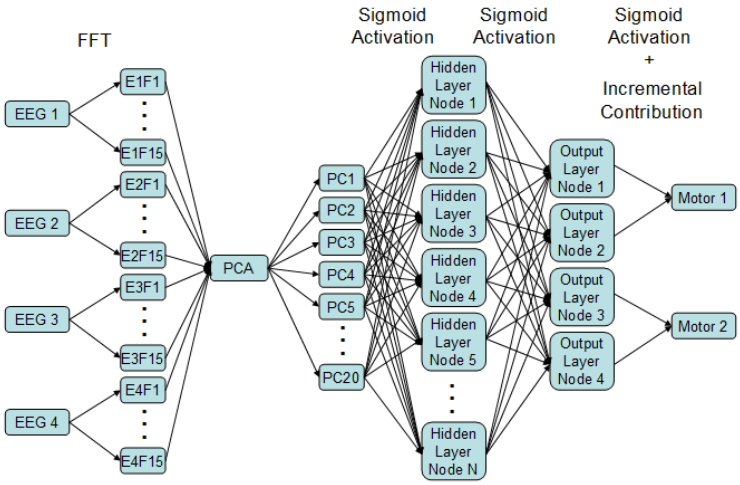


Figure 5.6: PCA - Sigmoid Activation Neural Network Control System Diagram

This C++ program begins by loading data from a training file, a test file, a feature vector file, and a weightings file. This facilitates the continuation of neural network mapping that is already in progress. At this point user friendliness was dropped for developer convenience, so all file names are hard coded and in order to keep output files for records or to import past data sets, the files have to be re-named in their directory.

When the main menu loads, the following functions are available: Perform PCA on sample data, generate random weightings values, train neural network, test neural

network, and save neural network. Performing a PCA does not convert the sample data into principle components, but generates a new feature vector, which updates the variable space and writes a new feature vector file. Generating random weights creates new network weightings on the range [-1, 1] with 0.0001 increments.

Training the neural network is where the main focus of this program begins. An output file is generated using the current time, to which various data will be recorded during the training cycle. Until the necessary criteria are met, the function to adapt the neural network is called and the neural network is trained. Each loop outputs to the screen the previous accuracy observed, the RMSE on the output channels, and the current epoch.

Each epoch the program loops through the sample data, which is now stored in memory to improve speed. If the program is in PCA mode, the FFT magnitudes are transformed into the PCA space using the feature vector (Equation 5.19).

$$P^T = FV^T * F^T \tag{5.19}$$

Where P^T are the new principle component values, transposed, FV^T is the transpose of the feature vector, and F^T are the transpose of the FFT input values.

If the program is not in PCA mode, the FFT values are simply inserted into the PCA-values variable space to simplify the number of conditional statements required. Forward propagation now begins and in most implementations of this program, the input neurons use a sigmoid activation function as well (Equation 5.20). In one of the middle implementations, the sigmoid was removed from this step to test the effect of not passing the inputs through an activation function.

$$P_i = \frac{1}{1 + e^{-P_i}} \tag{5.20}$$

Next, the hidden layer values are accumulated (Equation 5.21) and then are modified by a sigmoid activation function (Equation 5.22).

$$H_i = \Sigma(P_{jk} * wh_{ijk}) \tag{5.21}$$

Where H_i are the hidden layer nodes, P_{jk} are the principle components or FFT inputs, and wh_{ijk} are the weightings from the input layer to the hidden layer.

$$H_i = \frac{1}{1+e^{-H_i}} \tag{5.22}$$

Finally, the output values are accumulated (Equation 5.23) and passed through a sigmoid activation function (Equation 5.24).

$$O_i = \Sigma(H_i * wo_{ij}) \tag{5.23}$$

Where O_i are the output layer neurons and wo_{ij} are the weightings from the hidden layer to the output layer.

$$O_i = \frac{1}{1+e^{-O_i}} \tag{5.24}$$

Here, a winner determination is performed for the sake of evaluating the nueral network's performance as a classifier, and if the outputs are within the accepted range of values set by a tolerance definition, the accuracy of the output is evaluated. A winner-takes-all routine, with its own evaluation of error was also tested, but won't be used until later. In the current implementation, the error is summed by the square of the difference between the outputs and their learning values (Equation 5.25).

$$E = \sum (T_i - O_i)^2 \tag{5.25}$$

Where E is the accumulated squared error on the output channels, T_i is the known output value, and O_i is the control system output value.

Backpropogation now begins with the addition of gradient descent, using the derivative of the sigmoid activation functions, as well as delta functions to enable learning momentum (Equations 5.26 through 5.30).

$$\Delta H_i = \sum O_j * (1 - O_j) * (T_j - O_j) * wo_{ij} \tag{5.26}$$

Where ΔH_i is the error at each hidden layer node and wo_{ij} is the hidden to output weightings.

$$wo_{ij} = wo_{ij} + (1 - M) * LR * O_j * (1 - O_j) * (T_j - O_j) * H_i + M * \Delta wo_{ij} \tag{5.27}$$

$$\Delta wo_{ij} = (1 - M) * LR * O_j * (1 - O_j) * (T_j - O_j) * H_i + M * \Delta wo_{ij} \tag{5.28}$$

Where M is the learning momentum factor, LR is the learning rate, and Δwo_{ij} is the rate of change of the hidden to output weightings.

$$wh_{ij} = wh_{ij} + (1 - M) * LR * H_i * (1 - H_i) * \Delta H_i * P_j + M * \Delta wh_{ij} \tag{5.29}$$

$$\Delta wh_{ij} = (1 - M) * LR * H_i * (1 - H_i) * \Delta H_i * P_j + M * \Delta wh_{ij} \tag{5.30}$$

Where wh_{ij} is the input to hidden layer weightings and Δwh_{ij} is the rate of change of the input to hidden layer weightings.

The accuracy is then printed to screen and the function returns the current level of error to the parent function, which prints the error to screen and continues looping if the conditions to stop have not been met. Testing the neural network runs the same forward propogation as the refinement function, but stops at evaluating the accuracy and error of the system without modifying the weightings.

Adaptation

This project specifically aims to provide artificial limb control for amputees. As a result of this requirement, the control system must have methods in place to improve performance when no optimal feedback is available. For this reason the system has been designed to adapt the control system based on NIRS data in real-time, to primarily mitigate error as well as improve system performance by more personalized brain-to-muscle mapping. NIRS is used to detect general brain activity and act as a switch to enable adaptation, mapping the current EEG inputs onto EMG outputs assumed to be the current state of the system based on the control algorithm output and NIRS information.

Fuzzy-Neighbors Adaptation

The current adaptation routine used for the fuzzy neighbors algorithm is overly simplified because it needs to see clinical trials before real progress can be made. If the right side sensor determines that a significant amount of activity is occurring on the right side of the brain compared to the other two sensor regions, then counter indices are incremented or decremented to track how often a FFT index is active during this state, separated to correspond with each training set, based on the activity that is determined to be the current output of the system (Equations 5.31). If the left side of the brain is active during a left side of the body output activity, then the opposite values are incremented and decremented (Equations 5.32).

$$\begin{aligned} &if\,(N_a > N_u + 0.1 * N_m) \\ &A_{ij} = A_{ij} + F_j \end{aligned} \tag{5.31}$$

$$if\,(N_a > N_u + 0.1 * N_m)$$
$$A_{ij} = A_{ij} - F_j \tag{5.32}$$

Where N_a is the active NIRS sensor, N_u is the inactive NIRS sensor, N_m is the midline NIRS sensor, A_{ij} is the adaptation matrix, and F_j is the current FFT logic values.

If any of the counters exceed a positive or negative threshold, then the training matrix corresponding to that counter can be set to zero or one based on the threshold reached (Equations 5.33 and 5.34). In this way, as long as the output of the system is correct most of the time, the FFT indices that are generally active during that activity will impact the training set and slowly improve the consistency of the system.

$$if\,(Aij > Threshold)$$
$$Lj = 1 \tag{5.33}$$

$$if\,(Aij < -Threshold)$$
$$Lj = 0 \tag{5.34}$$

Where L_j indicates the Logic index for a known neighbor.

Neural Network Adaptation

The neural network training algorithms previously discussed have both used the adaptation method of updating neural network weights. This was used for the purpose of being able to train the network using a real-time data feed. This also enables the network to continuously adapt based on sensor data during implementation. When EMG sensors are not being used, the NIRS sensors can be used to confirm or contradict the current output of the neural network and generalized outputs (zero or one) can be used to train the network in real-time based on the input data occurring at that time. This is particularly

useful if the user is purposefully repeating a specific activity and there is more confidence that the system is producing the correct output. However testing this technique requires extended periods of use and hinges on the size of the network being kept small enough to not inhibit real-time adaptation since back propagation becomes computationally intense for large numbers of hidden nodes.

CHAPTER SIX: EXPERIMENTAL RESULTS AND IMPLEMENTATION

Proof of Concept Artificial Limb

For the purpose of proof of concept as well as demonstrating the capability of the system, an arm was constructed to provide the physical output of the system in a basic manner. Three arm segments and two joints, elbow and wrist, were represented by modeling the approximate shape of an arm and making necessary adjustments to allow for actuation and motor mounting (Figures 6.1 – 6.3). The models were then input into a rapid prototyping machine and fabricated using ABS plastic. Servo motors were used to actuate the joints due to the low requirements of the initial system (data sheets available in Appendix A.7).

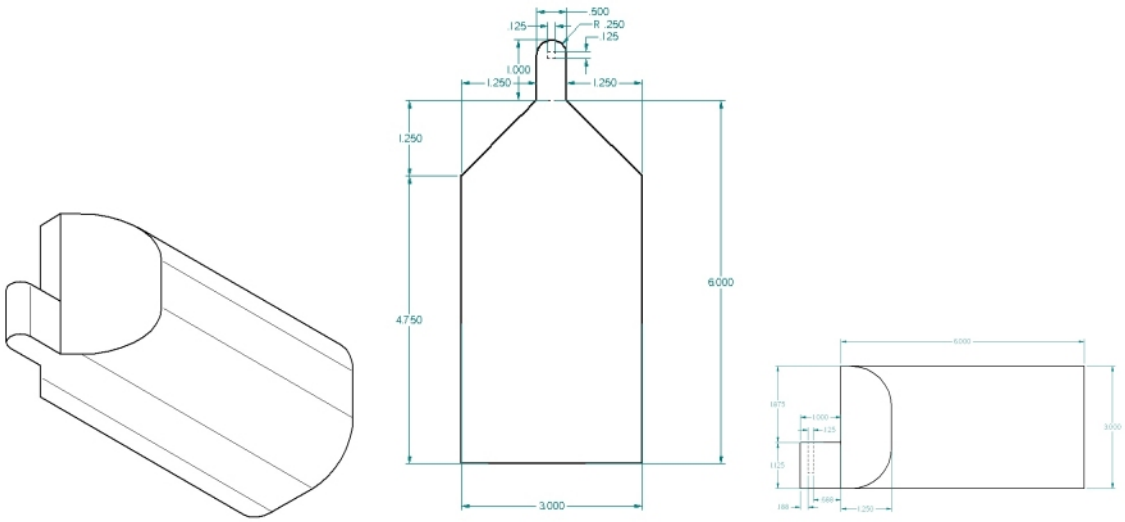


Figure 6.1 A-C: Upper Arm Modeling (Isometric (left), Top (center), and Side View (right))

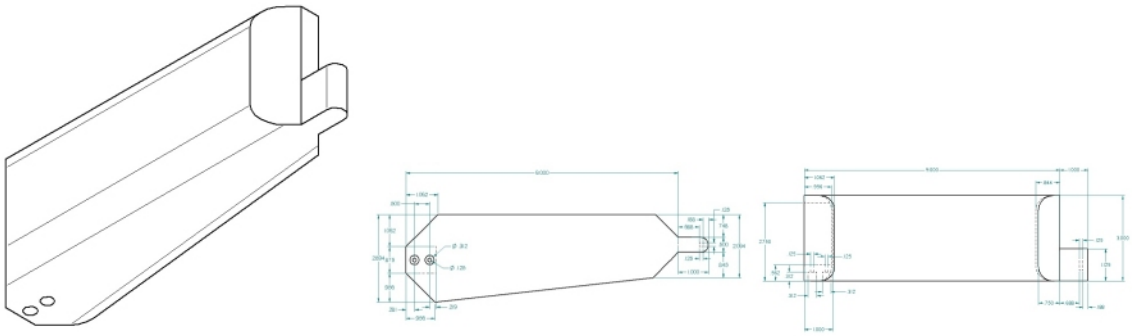


Figure 6.2 A-C: Forearm Modeling (Isometric (left), Top (center), and Side View (right))

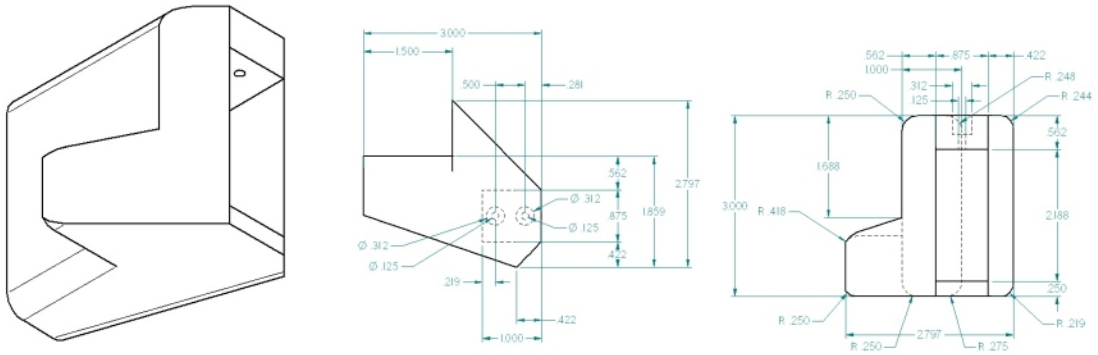


Figure 6.3 A-C: Hand Modeling (Isometric (left), Top (center), and Side View (right))

Since EMG values are used to train the control algorithms explored in this study, evaluating the effectiveness of each system is performed at the EMG output level and not at the physical output of the arm. Inconsistencies in approximating the appropriate level of arm flexion can arise from improper sensor placement or poor contact, causing the value observed on an EMG channel to not correspond to equal actuation on another. As a result, appropriate linear scaling must be performed on the EMG output values in order to be used for limb actuation.

Transfer Function

The motor control based on EEG was very similar to the simulated arm activity generated based on simultaneous recordings with EMG. Motor control in Figure 6.4

depicts movement of the elbow joint and full range of motion using the transfer function during five repetitions of an elbow flexion. Figure 6.5 indicates that the wrist was flexed as well during bicep curls, as reflected by simultaneous EMG readings indicating the secondary activity took place. In both graphs, the y-axis value represents change from the resting position, with 461 representing full proximal flexion of the elbow, 256 representing full proximal flexion of the wrist, and -256 representing full distal extension of the wrist.

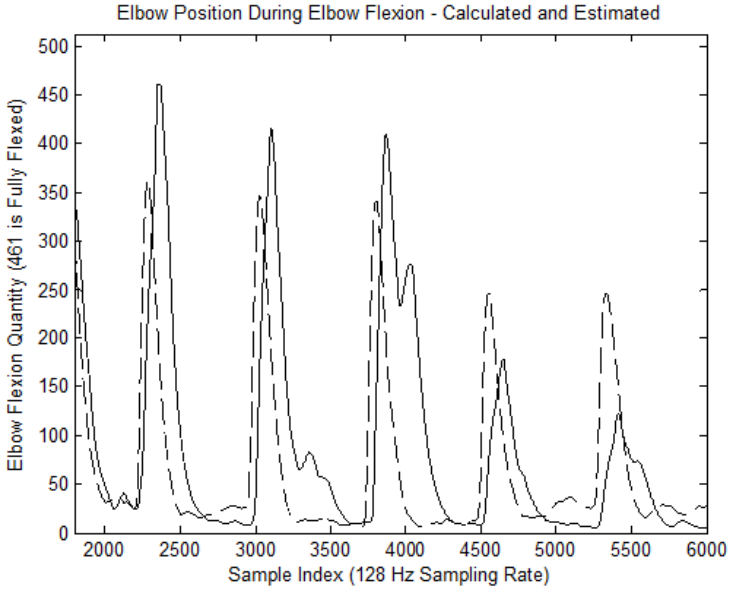


Figure 6.4: EMG Driven elbow activity (Dashed) and calculated values (Solid) during elbow flexion

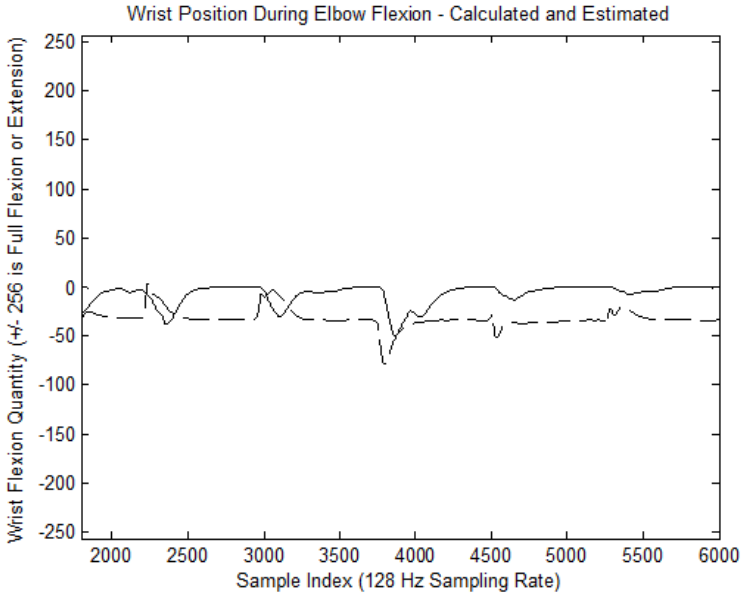


Figure 6.5: EMG Driven wrist activity (Dashed) and calculated values (Solid) during elbow flexion

There is a recurring time delay that appears in the correlation between calculated EMG and measured EMG data. This delay is shown in Figure 6.6, where the delay from peak of measured EMG activation to peak of calculated EMG activation averages 73 samples, or 0.57 seconds. This delay is primarily caused by the transfer function utilizing the last one second of data for input parameters, which leads to a slower reaction time.

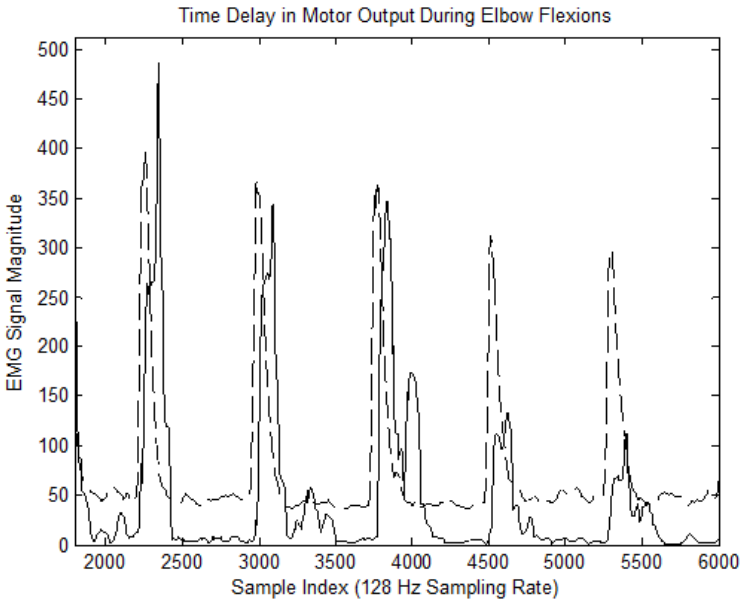


Figure 6.6: Monitored muscle activity (Dashed) and calculated values (Solid) during elbow flexion

Figure 6.7 shows the improvement in activity matching obtained through the difference/average method compared to the original individual channels method after the transfer function algorithm was configured to automatically determine system parameters. The major activity of the elbow flexion is nearly identical in both cases, but the resting position and sporadic motion in the second method are much improved.

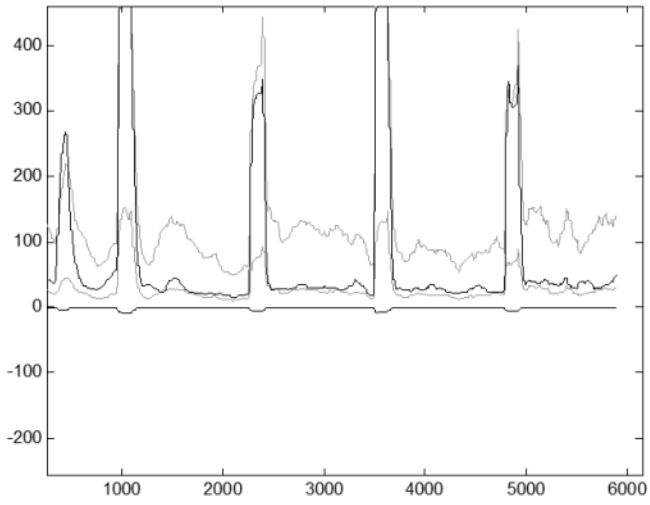


Figure 6.7: Motor position from original method (Grey) and difference/average method (Black)

Case Matching Algorithm

The second simple method of motor control, based on selection of a case using minimum error determination, produced less accurate results, but was still investigated as an alternative due to the higher degree of complexity possible using this system without moving to an advanced machine learning technique. The performance of the case selection algorithm is depicted in Figure 6.8 where a significant number of inaccurate movements can be seen and full range of motion was not achieved.

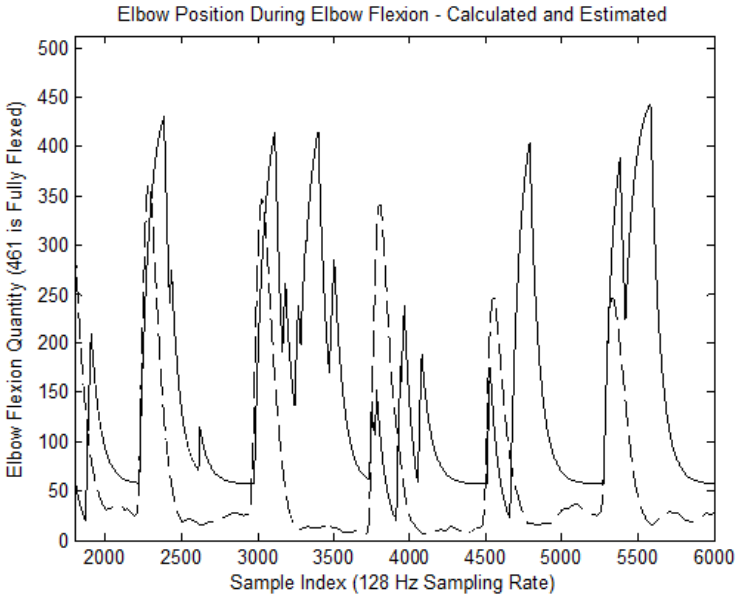


Figure 6.8: EMG Driven elbow (Dashed) and EEG case driven values (Solid) during elbow flexion

The case matching algorithm also produced a wrist flexion that is indicative of wrist tensing occurring during an elbow flexion (Figure 6.9), showing flexion of the wrist simultaneously occurring with elbow flexion as opposed to the minor wrist extension that the EMG data and transfer function calculations would suggest.

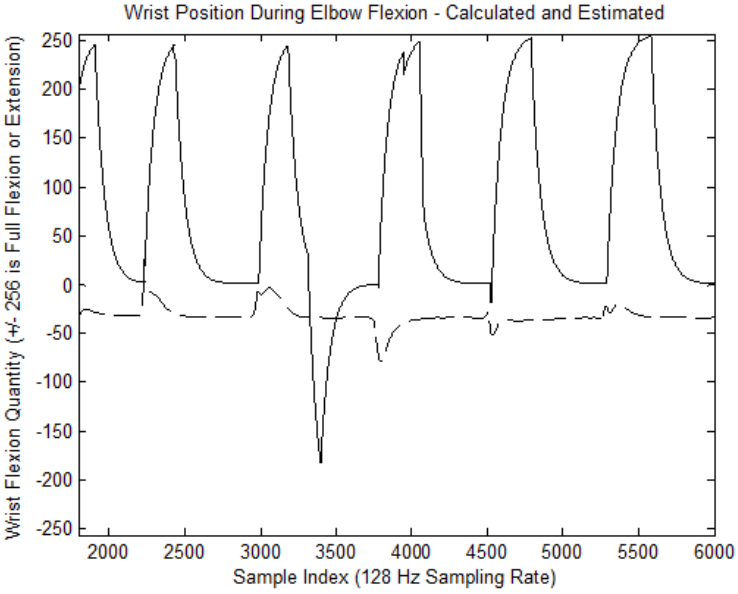


Figure 6.9: EMG Driven wrist (Dashed) and EEG case driven values (Solid) during elbow flexion

Fuzzy-Nearest Neighbors

Using 600 sample points to populate the known neighbor data, the fuzzy neighbors algorithm is able to self-evaluate with an accuracy of 100% (Figure 6.10) and a mean absolute error of less than 2%. However, this algorithm performs very poorly against unique test data (Figure 6.11). The algorithm is significantly over-fitted to the training data and does not consider most of the test data to be close enough to associate with the known neighbors. As a result, only two of the six hundred test points are recognized and correctly classified, while eight points are misclassified. The output estimation error for the "hit" cases is still very good, but the misses result in large deviations from the desired activity since they misidentified the correct output. Relaxing the determination criteria to reduce the number of test points classified as noise, causes the system to only classify input data correctly at the same rate as random chance (1 of 6) both against the known data as well as test data. Increasing the number of points in the known neighbors list to 1500 resulted in more stringent criteria in order for 100% accuracy against the training set, causing the test results to produce only one hit and zero misses.

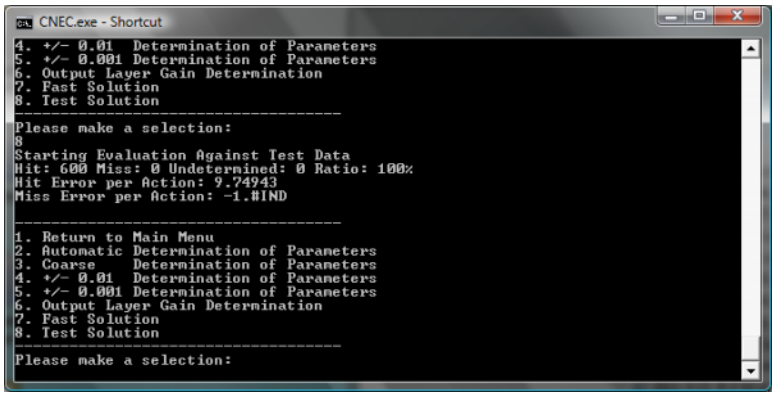


Figure 6.10: Fuzzy-neighbors accuracy against the training data set

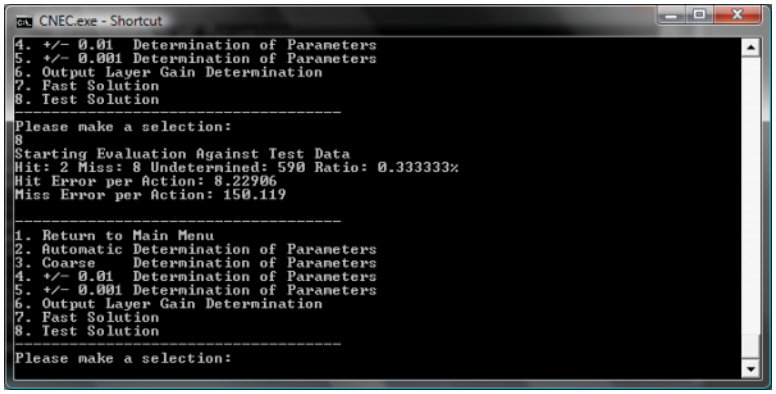


Figure 6.11: Fuzzy-neighbors accuracy against the test data set

k-Nearest Neighbor

Based on the results of the fuzzy-neighbors algorithm, larger data sets were prepared prior to testing the k-nearest neighbors program and the outputs of the system was reduced to the four training activities that have direct correlation to the EMG data. A training file with 21,547 data sets was used and 5- and 10-fold sampling was performed to randomly select a subset of the data to be withheld as the test set. Using various values of k, the accuracies in Figures 6.12 and 6.13 were obtained (details available in Tables 6.1 and 6.2).

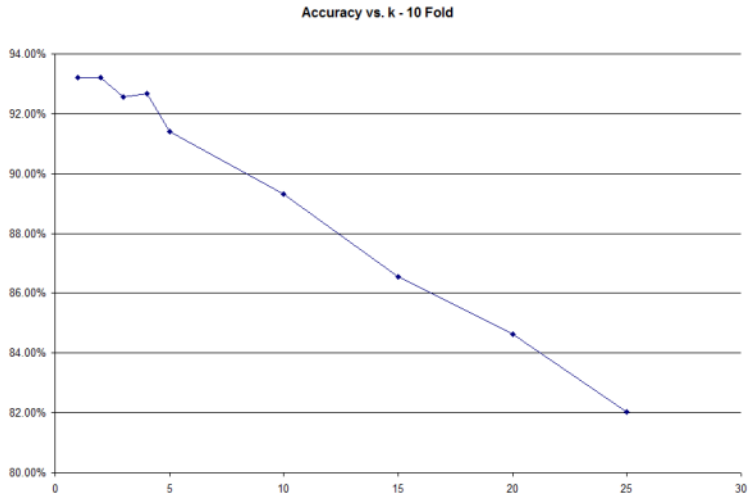


Figure 6.12: 10-Fold k-Nearest Neighbor Accuracy Using Various k Values

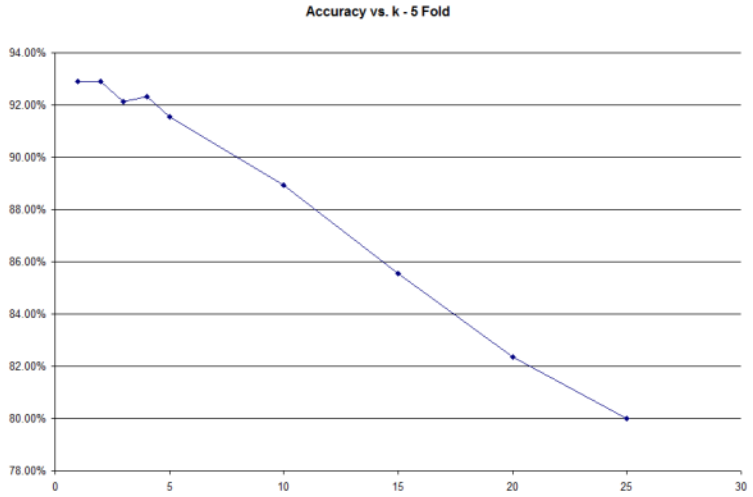


Figure 6.13: 5-Fold k-Nearest Neighbor Accuracy Using Various k Values

Table 6.1: 10-Fold k-Nearest Neighbor Performance at Various k Values

k	Accuracy	RMSE EMG 1	RMSE EMG 2	RMSE EMG 3	RMSE EMG 4	Average RMSE
1	93.22%	59.6497	37.849	18.9752	62.4031	44.71925
2	93.22%	62.2091	36.9362	17.5225	55.501	43.0422
3	92.57%	70.1476	38.405	18.2062	55.9462	45.67625
4	92.66%	75.8983	41.5508	20.814	60.9629	49.8065
5	91.41%	82.3058	44.5931	23.0636	65.2384	53.800225
10	89.32%	114.807	62.5845	29.6825	89.9904	74.2661
15	86.54%	138.01	73.8761	34.6716	106.248	88.201425
20	84.63%	155.636	83.1942	37.9804	117.15	98.49015
25	82.03%	167.458	90.2292	40.7795	125.356	105.955675

Table 6.2: 5-Fold k-Nearest Neighbor Performance at Various k Values

k	Accuracy	RMSE EMG 1	RMSE EMG 2	RMSE EMG 3	RMSE EMG 4	Average RMSE
1	92.90%	66.4866	34.1634	18.6023	63.7847	45.75925
2	92.90%	66.6687	35.9309	18.522	57.9707	44.773075
3	92.13%	72.7172	39.4172	20.1938	63.3027	48.907725
4	92.32%	80.6528	43.5965	22.819	68.2781	53.8366
5	91.55%	89.044	48.8127	24.5912	73.547	58.998725
10	88.93%	122.477	68.0891	32.0496	98.0114	80.156775
15	85.56%	146.005	80.1682	36.6322	114.208	94.25335
20	82.36%	162.064	89.153	39.8106	124.604	103.9079
25	79.99%	173.79	95.6071	42.2086	132.297	110.975675

Selecting a k of two produces the least root mean square error while tying for best accuracy. At 93.22% accuracy and 43.04 (approximately 8.41%) RMSE (10-fold), this algorithm satisfies the design requirements. Unfortunately the fact that the algorithm performs best with a very low k and quickly degrades as k increases, indicates that the classification algorithm sees a lot of overlap between the output classes. While investigating how badly this overlap could affect real-time performance, it was determined that using a completely unique dataset instead of reserving a portion of the training set, results in an accuracy around 41%, which does not meet the requirements of this project and is too great a decrease in performance to rely on sensor improvements.

Linear Activation Neural Network

The linear activation neural network was able to train up to a convergence of 891 out of 1000 training values being within tolerance. However after this point the learning process went unstable and the error went to infinity (Figure 6.14).

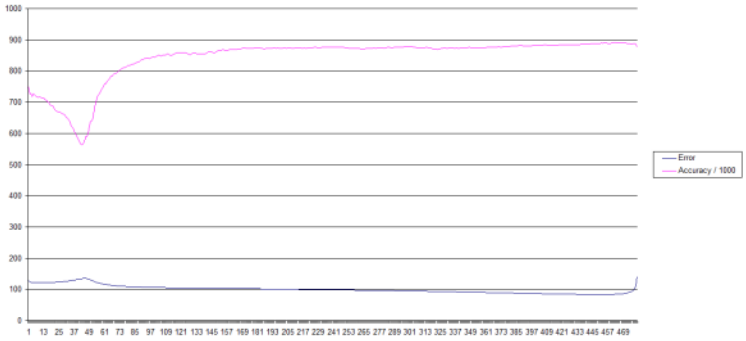


Figure 6.14: Error and Accuracy of Linear Activation Neural Network – Diverges

Principle Component Analysis

Using PCA on this data set did result in good reduction of dimensionality, however it did not produce obvious distinction between output activities (Figures 6.15 – 6.18).

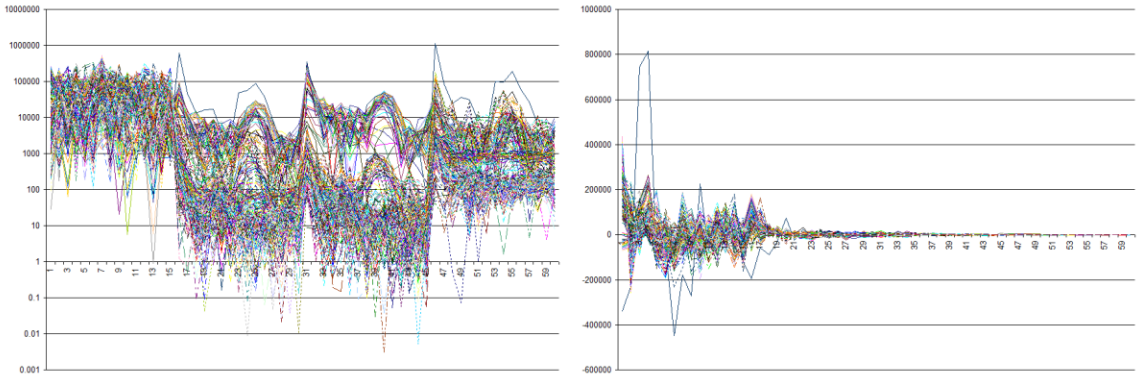


Figure 6.15 A&B: 250 Bicep Flexions - FFT (Left) and PCA (Right)

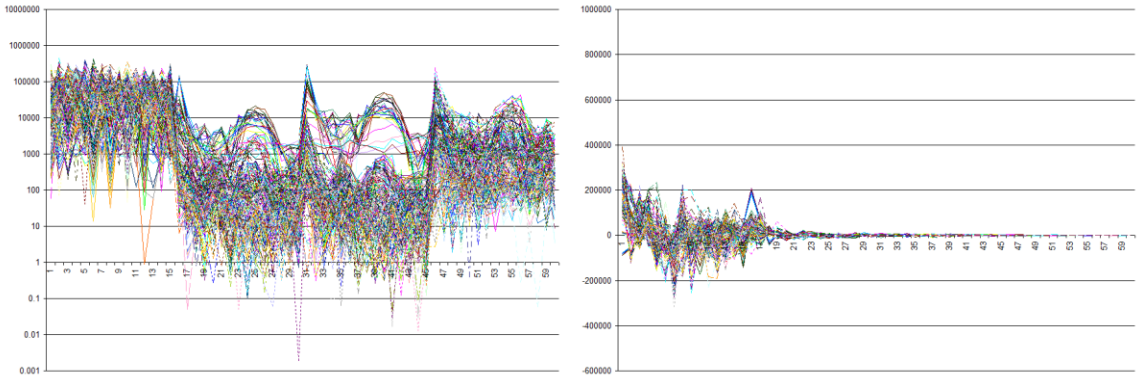


Figure 6.16 A&B: 250 Triceps Extensions - FFT (Left) and PCA (Right)

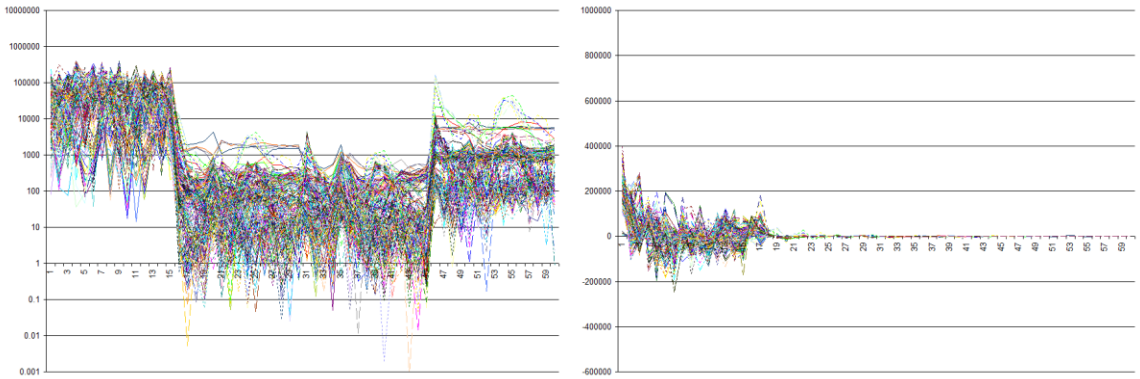


Figure 6.17 A&B: 250 Wrist Flexions - FFT (Left) and PCA (Right)

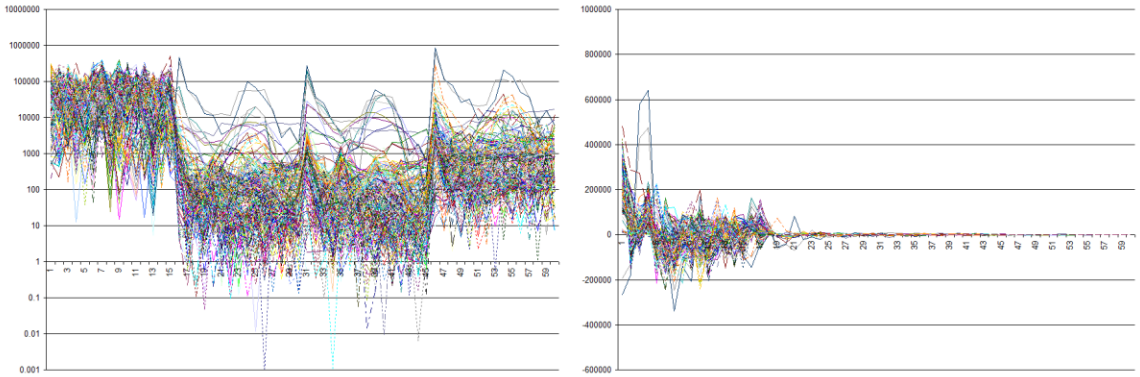


Figure 6.18 A&B: 250 Wrist Extensions - FFT (Left) and PCA (Right)

Looking at the mean value of the data for each activity, there is some distinction, but not a significant amount compared to the deviation within each output type (Figure 6.19). The mean values clearly indicate that Eigenvectors passed the first twenty can be disregarded without substantial loss of data.

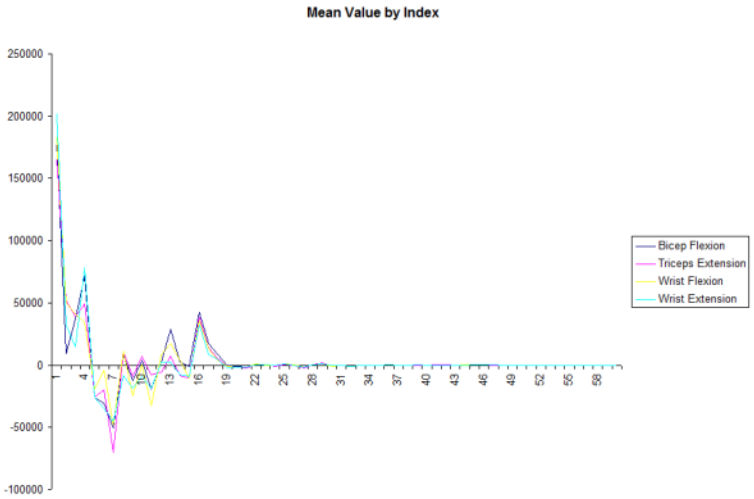


Figure 6.19: Mean PCA Indices by Activity

Sigmoid Activation Neural Network

SANN – Regression

To verify the validity of the training routine, the system was first tested on a XOR data set. Using decreasing numbers of hidden nodes, the system was proven by its ability to train to perform the function XOR (Table 6.3). The network did have difficulty approaching convergence using only two hidden nodes and for the purpose of saving time, the test was ended at 256,000 epochs.

Table 6.3: Neural Network's Ability to Train to Perform the Function XOR

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Conv-erge	RMSE at Outputs	Test Results	Test RMSE of Network
4 (XOR)	PCA Off	2	2	0.3	0.05	0.1	at 256k 3/4	0.057695	75.00%	0.115457
4 (XOR)	PCA Off	2	3	0.3	0.05	0.1	26248	0.01285055	100.00%	0.0741506
4 (XOR)	PCA Off	2	4	0.3	0.05	0.1	18088	0.0088945	100.00%	0.0232723
4 (XOR)	PCA Off	2	5	0.3	0.05	0.1	16808	0.00930755	100.00%	0.0274043
4 (XOR)	PCA Off	2	6	0.3	0.05	0.1	12083	0.00908415	100.00%	0.0179394

4 (XOR)	PCA Off	2	7	0.3	0.05	0.1	20075	0.00900055	100.00%	0.0216949
4 (XOR)	PCA Off	2	8	0.3	0.05	0.1	12685	0.0094273	100.00%	0.0251209

Once the system was validated, systematic tests were performed to determine the ability to train on EEG to EMG mapping data. Tests began with small sample sizes and gradually increased, evaluating the minimum number of hidden nodes necessary for the network to converge on a solution, as well as how well the system could discern outputs when acting on test data (Tables 6.4 - 6.7).

Table 6.4: Node Requirements and Test Accuracy, 16 Sample Data Set, PCA Off

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
16	PCA Off	60	4	0.3	0.05	0.1	506	0.0252949	12.50%	0.54137
16	PCA Off	60	8	0.3	0.05	0.1	352	0.0299741	6.25%	0.601325
16	PCA Off	60	12	0.3	0.05	0.1	530	0.02163385	0.00%	0.560804
16	PCA Off	60	16	0.3	0.05	0.1	355	0.02797195	12.50%	0.564746
16	PCA Off	60	20	0.3	0.05	0.1	581	0.0202052	12.50%	0.594552
16	PCA Off	60	24	0.3	0.05	0.1	777	0.01796875	12.50%	0.596938
16	PCA Off	60	28	0.3	0.05	0.1	371	0.0213859	12.50%	0.590905
16	PCA Off	60	32	0.3	0.05	0.1	709	0.01892985	12.50%	0.545535
16	PCA Off	60	36	0.3	0.05	0.1	1074	0.01728575	6.25%	0.567855
16	PCA Off	60	40	0.3	0.05	0.1	441	0.0202997	25.00%	0.560401
16	PCA Off	60	44	0.3	0.05	0.1	237	0.02698245	12.50%	0.556159
16	PCA Off	60	48	0.3	0.05	0.1	461	0.0214846	18.75%	0.361409
16	PCA Off	60	52	0.3	0.05	0.1	277	0.0241053	6.25%	0.368648
16	PCA Off	60	56	0.3	0.05	0.1	330	0.02848525	6.25%	0.383947
16	PCA Off	60	60	0.3	0.05	0.1	513	0.0218347	12.50%	0.603481

Table 6.5: Node Requirements and Test Accuracy, 16 Sample Data Set, PCA On

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
16	PCA On	20	4	0.3	0.05	0.1	663	0.02860755	12.50%	0.686654
16	PCA On	20	8	0.3	0.05	0.1	361	0.0297086	12.50%	0.648444
16	PCA On	20	12	0.3	0.05	0.1	431	0.0230049	12.50%	0.624149
16	PCA On	20	16	0.3	0.05	0.1	214	0.0351963	18.75%	0.640722
16	PCA On	20	20	0.3	0.05	0.1	225	0.0315515	25.00%	0.652475
16	PCA On	20	24	0.3	0.05	0.1	181	0.0275096	18.75%	0.647174
16	PCA On	20	28	0.3	0.05	0.1	282	0.02236195	12.50%	0.683215
16	PCA On	20	32	0.3	0.05	0.1	256	0.02612545	6.25%	0.613402
16	PCA On	20	36	0.3	0.05	0.1	163	0.038805	18.75%	0.316533
16	PCA On	20	40	0.3	0.05	0.1	507	0.01949595	12.50%	0.637014
16	PCA On	20	44	0.3	0.05	0.1	255	0.0281627	12.50%	0.639454
16	PCA On	20	48	0.3	0.05	0.1	292	0.02932515	12.50%	0.684582
16	PCA On	20	52	0.3	0.05	0.1	271	0.02094495	18.75%	0.655163
16	PCA On	20	56	0.3	0.05	0.1	252	0.0265533	18.75%	0.65766
16	PCA On	20	60	0.3	0.05	0.1	310	0.027524	6.25%	0.617368

Table 6.6: Node Requirements and Test Accuracy, 128 Sample Data Set, PCA Off

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
128	PCA Off	60	4	0.3	0.05	0.1	at 64k 71/128	0.0957255	10.94%	0.688465
128	PCA Off	60	8	0.3	0.05	0.1	at 64k 127/128	0.0241815	17.19%	0.689977
							at 128k 127/128	0.02017245	15.63%	0.684005
128	PCA Off	60	12	0.3	0.05	0.1	9417	0.02742745	14.06%	0.65792
128	PCA Off	60	16	0.3	0.05	0.1	14183	0.01705275	11.72%	0.645033

128	PCA Off	60	20	0.3	0.05	0.1	10068	0.01751325	16.41%	0.715138
128	PCA Off	60	24	0.3	0.05	0.1	10696	0.01566945	18.75%	0.662374
128	PCA Off	60	28	0.3	0.05	0.1	7136	0.02036165	21.09%	0.638219
128	PCA Off	60	32	0.3	0.05	0.1	8946	0.01524115	21.09%	0.631908
128	PCA Off	60	36	0.3	0.05	0.1	8229	0.01709495	17.97%	0.661223
128	PCA Off	60	40	0.3	0.05	0.1	7726	0.01892685	12.50%	0.655245
128	PCA Off	60	44	0.3	0.05	0.1	7891	0.0193626	14.06%	0.613598
128	PCA Off	60	48	0.3	0.05	0.1	8950	0.01789585	13.28%	0.653293
128	PCA Off	60	52	0.3	0.05	0.1	6850	0.021595	17.19%	0.654336
128	PCA Off	60	56	0.3	0.05	0.1	7823	0.02226915	17.19%	0.654868
128	PCA Off	60	60	0.3	0.05	0.1	7166	0.022419	15.63%	0.691254

Table 6.7: Node Requirements and Test Accuracy, 128 Sample Data Set, PCA On

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
128	PCA On	20	4	0.3	0.05	0.1	at 64k 59/128	0.1455775	16.41%	0.508309
128	PCA On	20	8	0.3	0.05	0.1	at 64k 34/128	0.1567025	1.56%	0.5101
128	PCA On	20	12	0.3	0.05	0.1	at 64k 87/128	0.0859415	9.38%	0.59056
128	PCA On	20	16	0.3	0.05	0.1	at 64k 80/128	0.0949775	10.16%	0.564615
128	PCA On	20	20	0.3	0.05	0.1	at 64k 101/128	0.067057	19.53%	0.729581
128	PCA On	20	24	0.3	0.05	0.1	at 64k 86/128	0.0980025	9.38%	0.590972
128	PCA On	20	28	0.3	0.05	0.1	at 64k 88/128	0.0900715	10.16%	0.620793
128	PCA On	20	32	0.3	0.05	0.1	13085	0.02394395	14.84%	0.680476
128	PCA On	20	36	0.3	0.05	0.1	17056	0.0170701	21.88%	0.713674
128	PCA On	20	40	0.3	0.05	0.1	20893	0.0245298	14.84%	0.732772
128	PCA On	20	44	0.3	0.05	0.1	14016	0.01830545	7.81%	0.784155

128	PCA On	20	48	0.3	0.05	0.1	19950	0.0138206	17.19%	0.70848
128	PCA On	20	52	0.3	0.05	0.1	15008	0.01465245	6.25%	0.766267
128	PCA On	20	56	0.3	0.05	0.1	18193	0.022721	13.28%	0.716033
128	PCA On	20	60	0.3	0.05	0.1	15718	0.0143935	8.59%	0.72833

During the course of evaluating the effectiveness of this program, it was determined that the organization of the input data was having a negative impact on the results of training. For this reason a quick test to determine the effect of interlacing the data based on the output type was performed (Table 6.8).

Table 6.8: Results when using data separated by activity versus interlacing activities

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
512	PCA Off	60	60	0.3	0.05	0.1	54282	0.0163056	8.59%	0.726063
512_i	PCA Off	60	60	0.3	0.05	0.1	14039	0.019718	9.18%	0.694214
512	PCA On	60	60	0.3	0.05	0.1	at 64k 491/512	0.0447046	8.01%	0.583519
512_i	PCA On	60	60	0.3	0.05	0.1	at 64k 220/512	0.1366945	10.35%	1.1716

Based on these results, interlacing activities was determined to be beneficial to convergence speed. However, it did raise a question regarding why the interlacing appeared to cause a negative impact on PCA rotated data. The full round of 512 sample sets was performed using the interlaced data sets (Tables 6.9 and 6.10).

Table 6.9: Node requirements and test accuracy, 512 interlaced sample data set, PCA Off

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
512	PCA Off	60	36	0.3	0.05	0.1	at 64k 508/512	0.02225425	10.74%	0.71154
512	PCA Off	60	40	0.3	0.05	0.1	at 64k 510/512	0.0196367	11.33%	0.700633
							at 128k 511/512	0.017018	12.11%	0.708363

512	PCA Off	60	44	0.3	0.05	0.1	41956	0.0203068	10.55%	0.687881
512	PCA Off	60	48	0.3	0.05	0.1	61891	0.0157582	12.11%	0.686533
512	PCA Off	60	52	0.3	0.05	0.1	22492	0.017743	9.57%	0.718004
512	PCA Off	60	56	0.3	0.05	0.1	56154	0.0126281	10.55%	0.703695
512	PCA Off	60	60	0.3	0.05	0.1	14039	0.019718	9.18%	0.694214

Table 6.10: Node requirements and test accuracy, 512 interlaced sample data set, PCA On

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
512	PCA On	20	52	0.3	0.05	0.1	at 64k 241/512	0.121877	13.28%	0.601264
512	PCA On	20	56	0.3	0.05	0.1	at 64k 256/512	0.1208905	10.35%	0.685636
512	PCA On	20	60	0.3	0.05	0.1	at 64k 220/512	0.1366945	10.35%	1.1716

The next data preparation to be tested was a random sorting algorithm that would arrange the data randomly when it is being read from file. At this stage it was also decided to introduce non-active data points into our training set and to attempt to address the issue with PCA performance by removing the sigmoid activation function from the input nodes. New tests were performed with data set sizes of 250 and 500 (Tables 6.11 - 6.14).

Table 6.11: Node requirements and test accuracy, 250 random sample data set, PCA Off

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
250	PCA Off	60	8	0.3	0.05	0.1	at 64k 232/250	0.0179596	15.70%	0.495839
250	PCA Off	60	12	0.3	0.05	0.1	9203	0.01050125	18.40%	0.534522
250	PCA Off	60	16	0.3	0.05	0.1	1669	0.0110981	12.50%	0.430706
250	PCA Off	60	20	0.3	0.05	0.1	3550	0.0056886	20.00%	0.491859
250	PCA Off	60	24	0.3	0.05	0.1	2519	0.00639345	17.50%	0.51624

250	PCA Off	60	28	0.3	0.05	0.1	771	0.01002135	13.30%	0.461856
250	PCA Off	60	32	0.3	0.05	0.1	1265	0.00772095	17.30%	0.497081

Table 6.12: Node requirements and test accuracy, 250 random sample data set, PCA On

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
250	PCA On	20	36	0.3	0.05	0.1	at 64k 162/250	0.0399322	20.70%	0.582438
250	PCA On	20	40	0.3	0.05	0.1	2353	0.00986735	9.70%	0.516233
250	PCA On	20	44	0.3	0.05	0.1	947	0.0115815	18.20%	0.515848
250	PCA On	20	48	0.3	0.05	0.1	1397	0.01052875	17.00%	0.583838
250	PCA On	20	52	0.3	0.05	0.1	1113	0.0098387	9.90%	0.469767
250	PCA On	20	56	0.3	0.05	0.1	2459	0.0090968	4.90%	0.483781
250	PCA On	20	60	0.3	0.05	0.1	1705	0.0068548	9.50%	0.573049

Table 6.13: Node requirements and test accuracy, 500 random sample data set, PCA Off

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
500	PCA Off	60	16	0.3	0.05	0.1	at 64k 409/500	0.02245355	13.80%	0.482698
500	PCA Off	60	20	0.3	0.05	0.1	at 64k 494/500	0.0117248	15.00%	0.488333
500	PCA Off	60	24	0.3	0.05	0.1	27200	0.0094824	23.70%	0.660694
500	PCA Off	60	28	0.3	0.05	0.1	12267	0.0078672	19.40%	0.552097
500	PCA Off	60	32	0.3	0.05	0.1	4017	0.00816775	18.40%	0.524085

Table 6.14: Node requirements and test accuracy, 500 random sample data set, PCA On

Sample Data	Mode	Input Nodes	Hidden Nodes	Learning Rate	Learning Momentum	Tolerance +/-	Epochs to Converge	RMSE at Outputs	Test Results	Test RMSE of Network
500	PCA On	20	60	0.3	0.05	0.1	at 64k 208/500	0.05632	11.80%	0.525189
500	PCA On	20	68	0.3	0.05	0.1	at 64k 172/500	0.0821835	0.30%	0.526952
500	PCA On	20	88	0.3	0.05	0.1	at 64k 439/500	0.019613	10.90%	0.537828

500	PCA On	20	100	0.3	0.05	0.1	at 64k 305/500	0.0340444	15.40%	0.611488
500	PCA On	20	104	0.3	0.05	0.1	at 64k 261/500	0.0454783	8.10%	0.534613
500	PCA On	20	108	0.3	0.05	0.1	24909	0.009536	7.90%	0.50397
500	PCA On	20	112	0.3	0.05	0.1	3187	0.0104671	18.60%	0.606821
500	PCA On	20	116	0.3	0.05	0.1	1348	0.00898635	17.30%	0.535481
500	PCA On	20	120	0.3	0.05	0.1	14306	0.0101927	8.80%	0.555371

The results from these tests do not meet the minimum requirements stipulated for success. They indicate that a very large neural network may be required in order to handle a sufficient number of data sets to train the system to handle a broad enough range of input combinations. They did however present the idea that the number of hidden nodes may not cause the system to become over-fitted, which was the understanding that these trials were based upon (Figure 6.20). This idea has been presented in literature (Lawrence 1997), but would carry more merit if the resulting networks in our trials performed well on test data. Plots to discern correlation between other factors are presented in Figures 6.21 - 6.24.

Figure 6.20: Plot of Network Performance vs. Number of Hidden Nodes

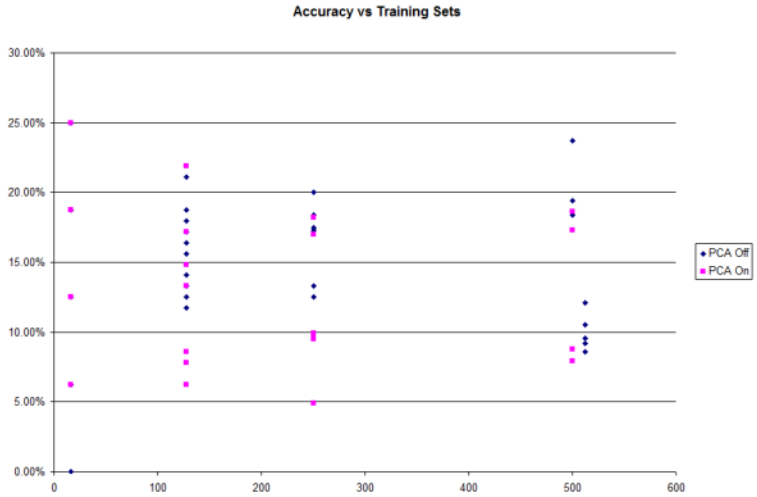


Figure 6.21: Plot of Network Performance vs. Training Sets

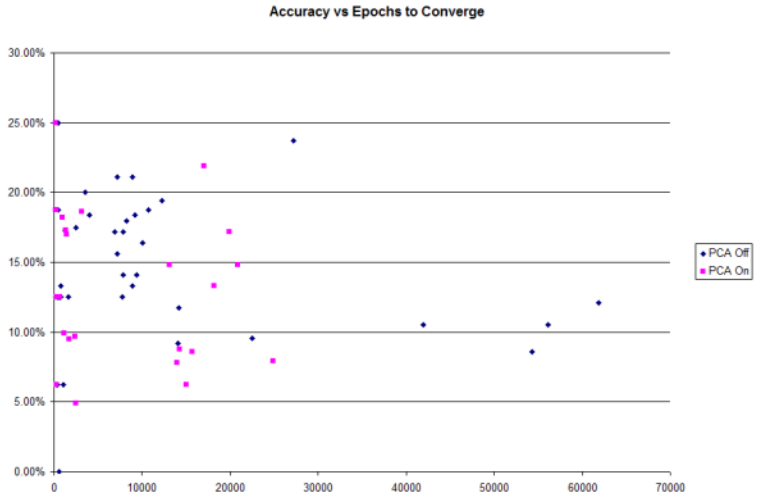


Figure 6.22: Plot of Network Performance vs. Epochs to Reach Convergence

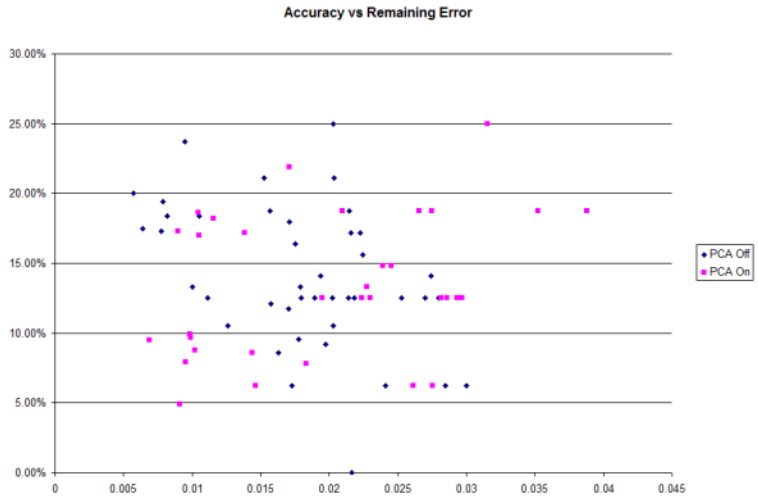


Figure 6.23: Plot of Network Performance vs. Remaining Training Error

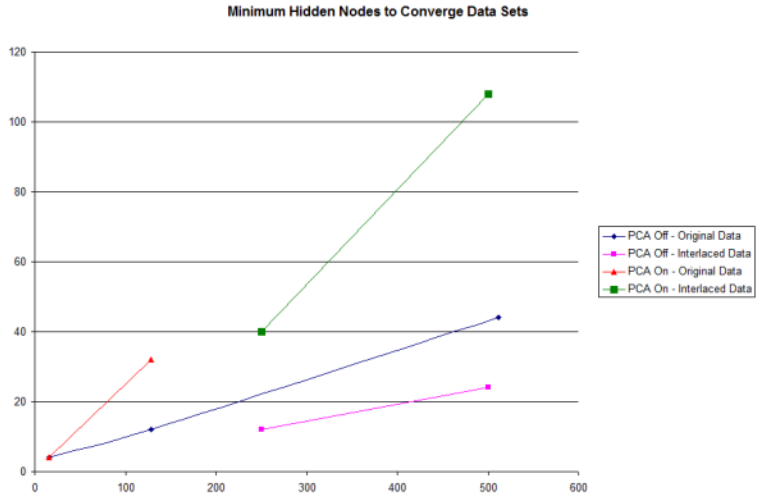


Figure 6.24: Plot of Minimum Hidden Nodes Required vs. Training Set Size

Based on the limited data in Figure 6.24, it can be concluded that less nodes are required when the data is not separated by activity. However, due to the apparent limits of using neural networks for regression on our data set, the program was adapted to simplify output data to perform classification without any regression capabilities.

SANN – Classification

As a classifier, the sigmoid activation neural network performs significantly better. The data set had to be changed in order to have a 'zero' activity class for the purpose of control and much larger data sets were used based on the previous results. Details of the tests performed are available in Tables 6.15 through 6.21 and summary graphs are available (Figures 6.25 – 6.29).

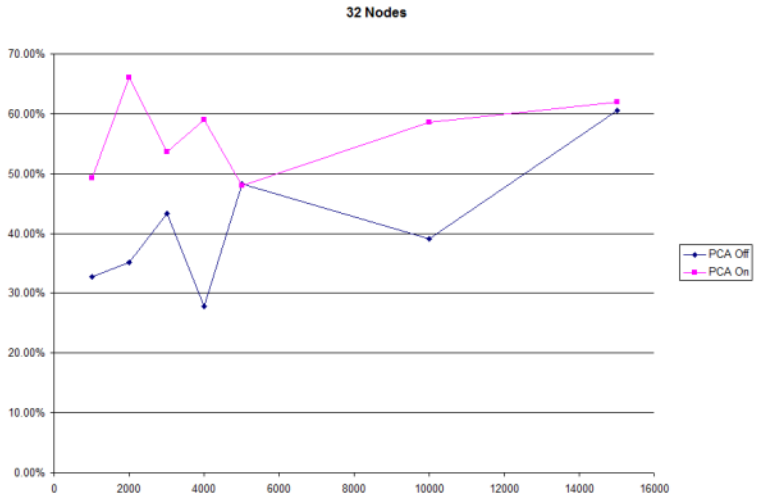


Figure 6.25: Test Accuracy vs. Data Set Size - 32 Hidden Nodes

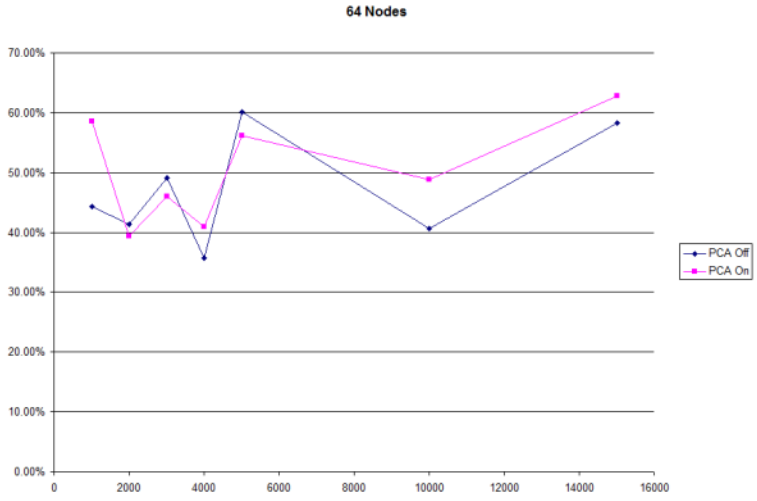


Figure 6.26: Test Accuracy vs. Data Set Size - 64 Hidden Nodes

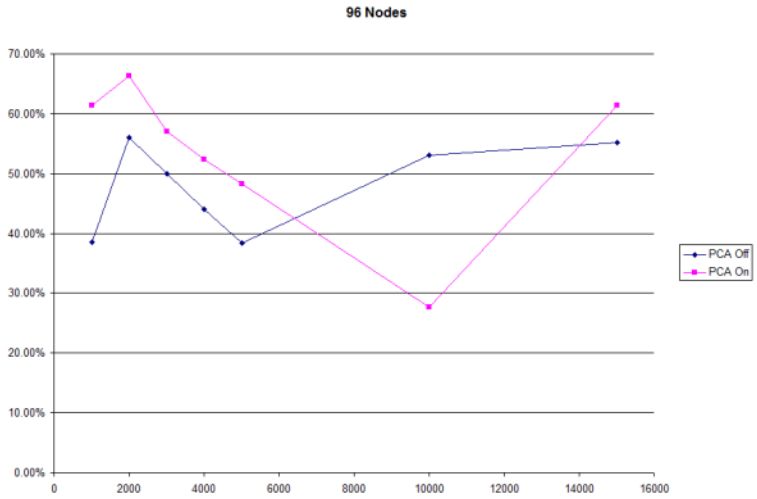


Figure 6.27: Test Accuracy vs. Data Set Size - 96 Hidden Nodes

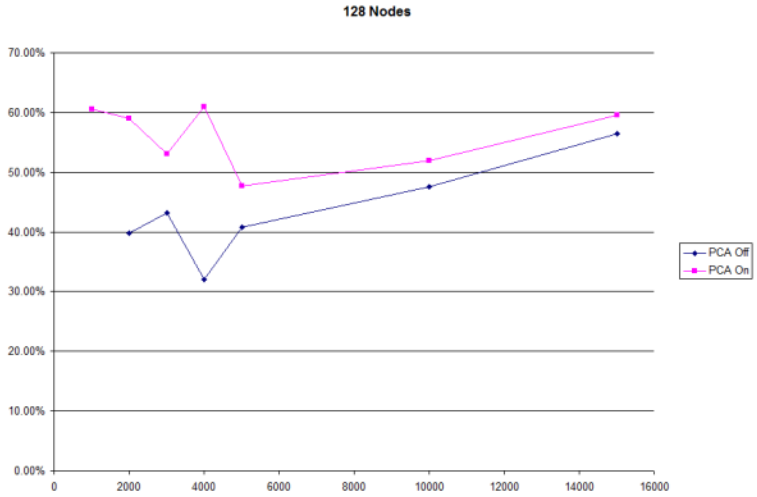


Figure 6.28: Test Accuracy vs. Data Set Size - 128 Hidden Nodes

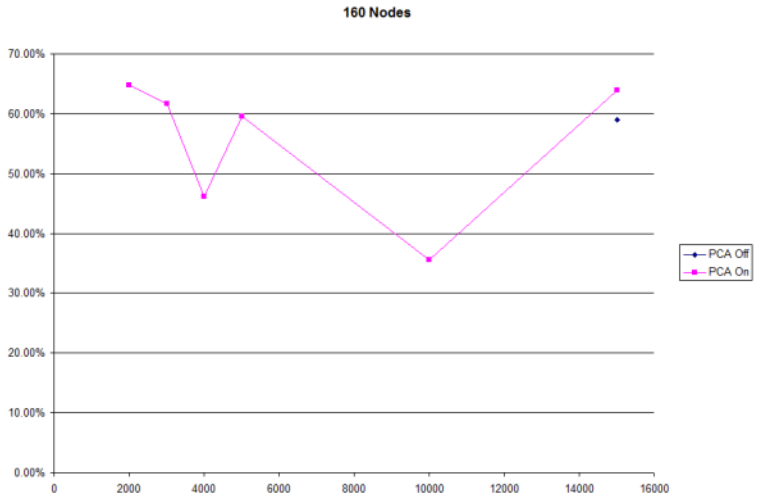


Figure 6.29: Test Accuracy vs. Data Set Size - 160 Hidden Nodes

Table 6.15: Test Accuracy at 1000 Epochs, 32 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA Off	32	1000	98.70%	0.0573653	32.70%	0.479829
2000	PCA Off	32	1000	95.60%	0.113127	35.20%	0.46527
3000	PCA Off	32	1000	97.27%	0.090427	43.30%	0.436045
4000	PCA Off	32	1000	96.93%	0.108605	27.80%	0.497991
5000	PCA Off	32	1000	94.74%	0.138549	48.20%	0.407709
10000	PCA Off	32	1000	89.30%	0.215385	39.10%	0.43711
15000	PCA Off	32	1000	88.14%	0.227197	60.60%	0.338521
Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA On	32	1000	96.10%	0.113386	49.30%	0.432822
2000	PCA On	32	1000	89.25%	0.218231	66.00%	0.378221
3000	PCA On	32	1000	88.30%	0.235057	53.60%	0.387018
4000	PCA On	32	1000	84.28%	0.266117	59.00%	0.436551
5000	PCA On	32	1000	82.86%	0.275933	48.00%	0.395032
10000	PCA On	32	1000	76.88%	0.297389	58.60%	0.339817
15000	PCA On	32	1000	77.80%	0.29042	61.90%	0.31503

Table 6.16: Test Accuracy at 1000 Epochs, 64 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA Off	64	1000	98.70%	0.0573426	44.30%	0.4423
2000	PCA Off	64	1000	95.95%	0.107351	41.30%	0.454376
3000	PCA Off	64	1000	95.40%	0.113765	49.10%	0.417351
4000	PCA Off	64	1000	97.10%	0.0964512	35.70%	0.463619
5000	PCA Off	64	1000	96.60%	0.102339	60.10%	0.356341
10000	PCA Off	64	1000	91.63%	0.188718	40.60%	0.431995
15000	PCA Off	64	1000	91.83%	0.193842	58.30%	0.353159
Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA On	64	1000	98.70%	0.0630732	58.60%	0.39385
2000	PCA On	64	1000	96.20%	0.133556	39.40%	0.457771
3000	PCA On	64	1000	92.20%	0.188385	46.00%	0.435881
4000	PCA On	64	1000	87.90%	0.212737	40.90%	0.432716
5000	PCA On	64	1000	85.14%	0.239868	56.20%	0.372251
10000	PCA On	64	1000	78.62%	0.290695	48.90%	0.386959
15000	PCA On	64	1000	87.20%	0.243771	62.80%	0.321029

Table 6.17: Test Accuracy at 1000 Epochs, 96 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA Off	96	1000	98.70%	0.0594825	38.50%	0.47078
2000	PCA Off	96	1000	96.50%	0.099492	56.00%	0.385545
3000	PCA Off	96	1000	97.27%	0.0952475	50.00%	0.430971
4000	PCA Off	96	1000	95.13%	0.1246985	44.10%	0.44073
5000	PCA Off	96	1000	93.98%	0.1313225	38.40%	0.454723
10000	PCA Off	96	1000	93.45%	0.168352	53.00%	0.383706
15000	PCA Off	96	1000	93.21%	0.177486	55.20%	0.360885
Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA On	96	1000	98.80%	0.0577674	61.40%	0.383691
2000	PCA On	96	1000	95.15%	0.117449	66.40%	0.332913
3000	PCA On	96	1000	89.93%	0.184791	57.00%	0.381057
4000	PCA On	96	1000	85.63%	0.231204	52.30%	0.393006
5000	PCA On	96	1000	82.76%	0.262383	48.20%	0.420999
10000	PCA On	96	1000	79.95%	0.270296	27.70%	0.479874
15000	PCA On	96	1000	78.33%	0.278724	61.40%	0.326498

Table 6.18: Test Accuracy at 1000 Epochs, 128 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA Off	128	1000	Does Not Train		Does Not Train	
2000	PCA Off	128	1000	99.25%	0.0490946	39.80%	0.441054
3000	PCA Off	128	1000	92.33%	0.15099	43.20%	0.449546
4000	PCA Off	128	1000	94.65%	0.126599	32.10%	0.524083
5000	PCA Off	128	1000	93.92%	0.141915	40.80%	0.447747
10000	PCA Off	128	1000	93.09%	0.16818	47.60%	0.410708
15000	PCA Off	128	1000	93.15%	0.177418	56.40%	0.351638
Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA On	128	1000	96.00%	0.104325	60.60%	0.377376
2000	PCA On	128	1000	93.20%	0.138139	59.00%	0.367192
3000	PCA On	128	1000	92.13%	0.155095	53.10%	0.413823
4000	PCA On	128	1000	87.45%	0.206029	61.00%	0.361258
5000	PCA On	128	1000	82.88%	0.245698	47.70%	0.414396
10000	PCA On	128	1000	75.48%	0.293127	52.00%	0.385874
15000	PCA On	128	1000	81.86%	0.257402	59.60%	0.340598

Table 6.19: Test Accuracy at 1000 Epochs, 160 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA Off	160	1000	Does Not Train		Does Not Train	
2000	PCA Off	160	1000	Does Not Train		Does Not Train	
3000	PCA Off	160	1000	Does Not Train		Does Not Train	
4000	PCA Off	160	1000	Does Not Train		Does Not Train	
5000	PCA Off	160	1000	Does Not Train		Does Not Train	
10000	PCA Off	160	1000	Does Not Train		Does Not Train	
15000	PCA Off	160	1000	91.08%	0.192462	59.00%	0.345252

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
1000	PCA On	160	1000	Does Not Train		Does Not Train	
2000	PCA On	160	1000	85.40%	0.194297	64.80%	0.344223
3000	PCA On	160	1000	88.37%	0.184796	61.70%	0.36259
4000	PCA On	160	1000	89.33%	0.188193	46.20%	0.442088
5000	PCA On	160	1000	88.64%	0.211418	59.50%	0.36919
10000	PCA On	160	1000	77.18%	0.287944	35.50%	0.438948
15000	PCA On	160	1000	80.83%	0.27604	63.90%	0.325038

Table 6.20: Test Accuracy at 1000 Epochs, 176 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
15000	PCA Off	176	150	Does Not Train		Does Not Train	
Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
15000	PCA On	176	1000	75.19%	0.283613	60.80%	0.331048

Table 6.21: Test Accuracy at 1000 Epochs, 192 Hidden Nodes

Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
15000	PCA Off	192	176	Does Not Train		Does Not Train	
Training Sets	Mode	Hidden Nodes	Epochs	Training Accuracy	Training RMSE	Test Accuracy	Test RMSE
15000	PCA On	192	110	Does Not Train		Does Not Train	

These results are much more promising than the regression neural network. The test sample, which has very different data from the training set, is demonstrating that neural networks are not always well suited to changing real-world conditions. Still a 66.4% accuracy rate through changing sensor conditions is promising, but not sufficient to conclude this technique is sufficient for use in this application. For this reason, a new version of the program was prepared that uses K-folding to train and test on different samples from the same data set, which will represent ideal operating performance.

SANN – K-Fold Classification

Using K-Folding to test on data that is not substantially different from the training set has produced very promising results. Details of the test results are available in Tables 6.22 and 6.23 and a summary of the results are available in Figures 6.30 and 6.31.

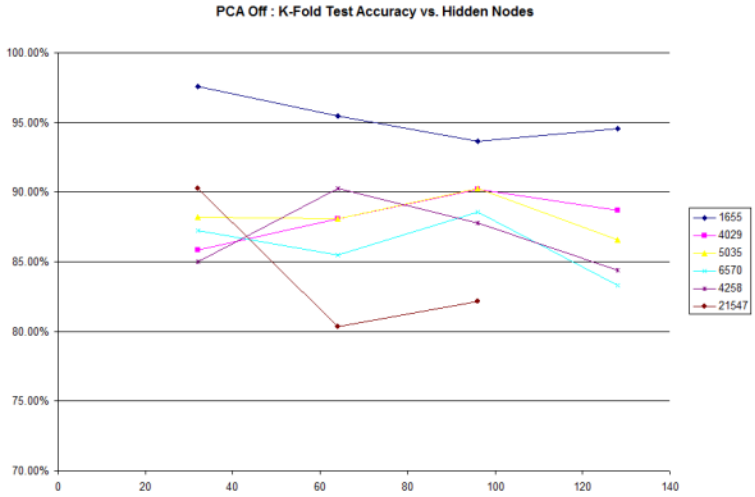


Figure 6.30: K-Fold Test Accuracy vs. Number of Hidden Nodes, PCA Off

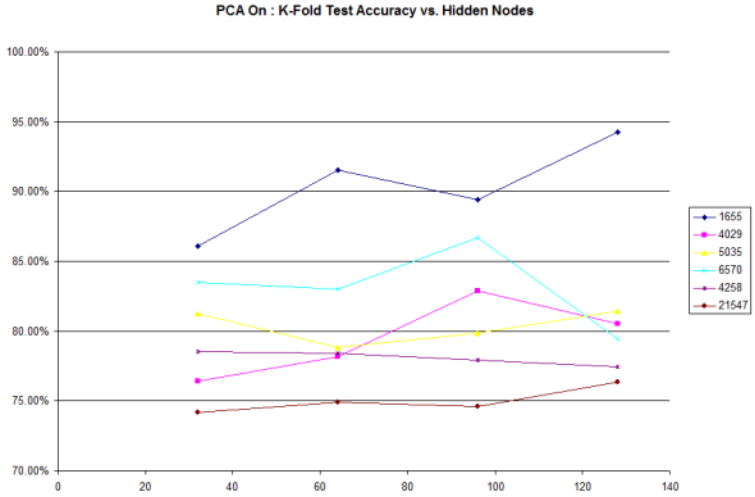


Figure 6.31: K-Fold Test Accuracy vs. Number of Hidden Nodes, PCA On

Table 6.22: K-Fold Test Accuracy at 1000 Epochs, PCA Off

Data Set	K-Fold	Mode	Hidden Nodes	Epochs	Learning Accuracy	Test Accuracy	Gap
1655	5	PCA Off	32	1000	97.96%	97.58%	-0.38%
1655	5	PCA Off	64	1000	96.00%	95.47%	-0.53%
1655	5	PCA Off	96	1000	96.07%	93.66%	-2.42%
1655	5	PCA Off	128	1000	95.62%	94.56%	-1.06%
4029	5	PCA Off	32	1000	94.17%	85.86%	-8.31%
4029	5	PCA Off	64	1000	95.53%	88.09%	-7.44%
4029	5	PCA Off	96	1000	96.93%	90.20%	-6.73%
4029	5	PCA Off	128	1000	94.45%	88.71%	-5.74%
5035	5	PCA Off	32	1000	94.66%	88.18%	-6.48%
5035	5	PCA Off	64	1000	96.28%	88.08%	-8.19%
5035	**5**	**PCA Off**	**96**	**1000**	**95.26%**	**90.27%**	**-4.99%**
5035	5	PCA Off	128	1000	90.09%	86.59%	-3.50%
6570	5	PCA Off	32	1000	92.71%	87.21%	-5.50%
6570	5	PCA Off	64	1000	93.38%	85.46%	-7.91%
6570	5	PCA Off	96	1000	91.04%	88.58%	-2.45%
6570	5	PCA Off	128	1000	87.96%	83.33%	-4.62%
4258	5	PCA Off	32	1000	95.24%	84.98%	-10.27%
4258	5	PCA Off	64	1000	94.63%	90.26%	-4.37%
4258	5	PCA Off	96	1000	93.04%	87.79%	-5.25%
4258	5	PCA Off	128	1000	88.67%	84.39%	-4.28%
21547	5	PCA Off	32	1000	94.63%	90.26%	-4.37%
21547	5	PCA Off	64	1000	87.00%	80.35%	-6.65%
21547	5	PCA Off	96	1000	89.25%	82.16%	-7.09%
21547	5	PCA Off	128	1000			
				Average:	93.50%	88.35%	-5.15%
				Std Dev:	3.05%	4.22%	2.61%

Table 6.23: K-Fold Test Accuracy at 1000 Epochs, PCA On

Data Set	K-Fold	Mode	Hidden Nodes	Epochs	Learning Accuracy	Test Accuracy	Gap
1655	5	PCA On	32	1000	91.31%	86.10%	-5.21%
1655	5	PCA On	64	1000	95.32%	91.54%	-3.78%
1655	5	PCA On	96	1000	91.92%	89.43%	-2.49%
1655	5	PCA On	128	1000	96.98%	94.26%	-2.72%
4029	5	PCA On	32	1000	78.03%	76.43%	-1.61%
4029	5	PCA On	64	1000	81.45%	78.16%	-3.28%
4029	5	PCA On	96	1000	91.90%	82.88%	-9.02%
4029	5	PCA On	128	1000	90.10%	80.52%	-9.58%
5035	5	PCA On	32	1000	89.50%	81.23%	-8.27%
5035	5	PCA On	64	1000	85.77%	78.85%	-6.93%
5035	5	PCA On	96	1000	84.43%	79.84%	-4.59%
5035	5	PCA On	128	1000	90.07%	81.43%	-8.64%
6570	5	PCA On	32	1000	86.76%	83.49%	-3.27%
6570	5	PCA On	64	1000	86.13%	83.03%	-3.10%
6570	**5**	**PCA On**	**96**	**1000**	**89.23%**	**86.68%**	**-2.55%**
6570	5	PCA On	128	1000	83.92%	79.45%	-4.47%
4258	5	PCA On	32	1000	79.68%	78.52%	-1.16%
4258	5	PCA On	64	1000	81.77%	78.40%	-3.36%
4258	5	PCA On	96	1000	83.41%	77.93%	-5.48%
4258	5	PCA On	128	1000	79.33%	77.46%	-1.87%
21547	5	PCA On	32	1000	77.09%	74.20%	-2.89%
21547	5	PCA On	64	1000	79.05%	74.92%	-4.13%
21547	5	PCA On	96	1000	77.54%	74.57%	-2.97%
21547	5	PCA On	128	1000	79.74%	76.36%	-3.38%
				Average:	85.43%	81.07%	-4.36%
				Std Dev:	5.87%	5.31%	2.42%

Implementation

The final C++ program for implementation uses the components from several of the programs used during testing (Figure 6.32). Basic communication with the sensor system and artificial limb is accomplished using the methods in chapter three. A start-up menu is presented to the user immediately since no pre-loading of data is necessary with this system design. From the start-up menu the following options are available: Record Baseline, Conduct Training Session, or Record Data.

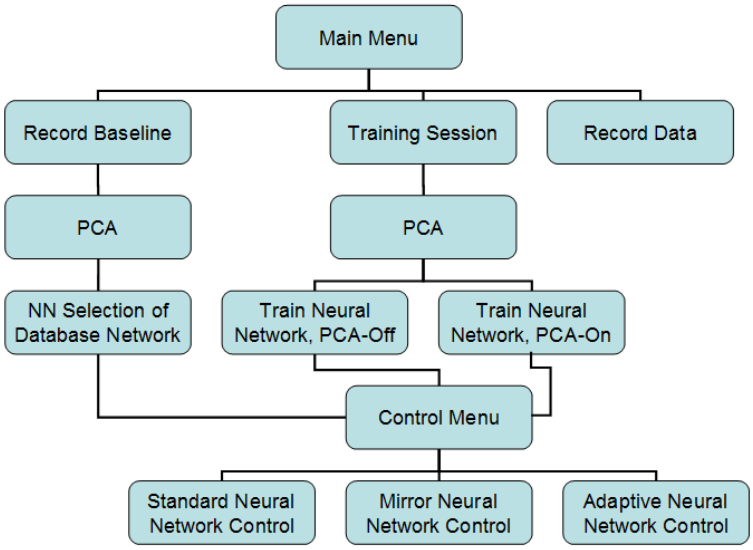


Figure 6.32: Flow Diagram of Implemented Training and Control System

If the user selects record baseline, the program steps through recording sessions for each activity to collected EEG data (Figure 6.33), while storing FFT data in memory (Figure 6.34). After the baseline is completed, the mean and variance of each channel is calculated and a principle component analysis if performed to generate the feature vector (Figure 6.35). After this function is completed, a nearest neighbor algorithm is used to select the closest matching pre-trained neural network from the database, using the weighted-Cartesian distance between feature vector indices to select the closest match (Figure 6.36). The weighting applied to the distance is distributed such that each subsequent Eigenvector has half as much influence as the previous Eigenvector. This is used to include the significance of the descending principle components into the nearest neighbor decision. Once the closest matching feature vector is selected, the corresponding neural network is loaded into the variable space along with the session ID associated with the pre-trained network, which is printed to the screen so that the operator

knows what has been loaded. If the loaded neural network uses PCA, that setting is also loaded into the control system. At this time the real-time control menu appears.

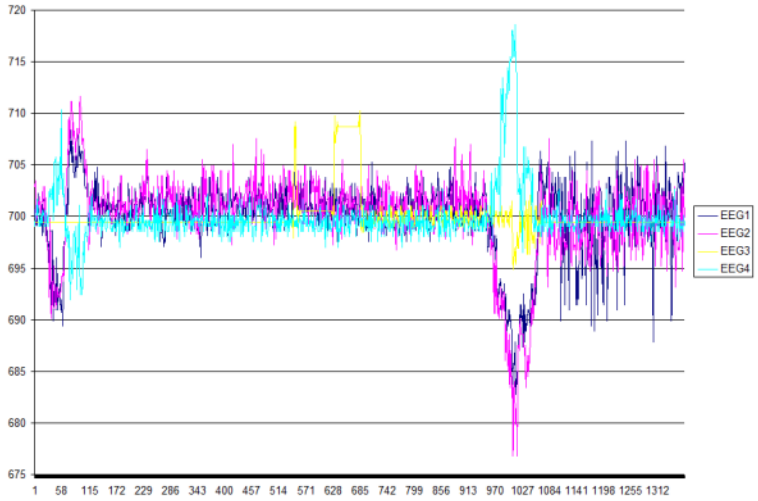


Figure 6.33: Raw EEG Data Collected During Baseline or Training Sessions

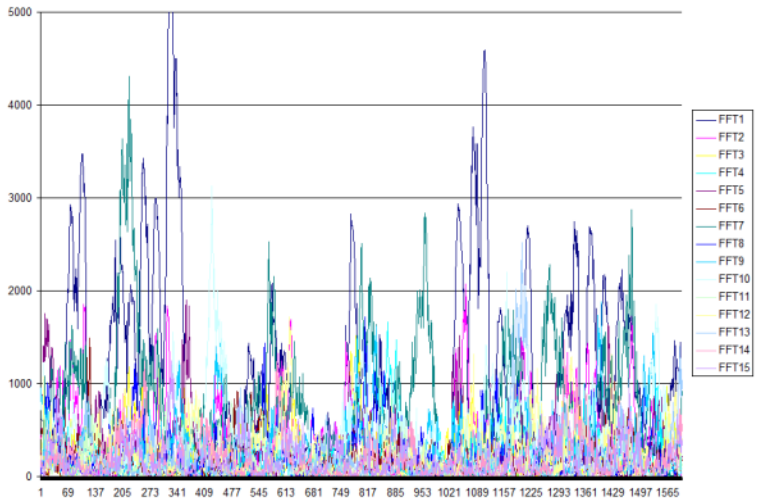


Figure 6.34: FFT of EEG Data, Stored Into Memory for PCA or Neural Network Use

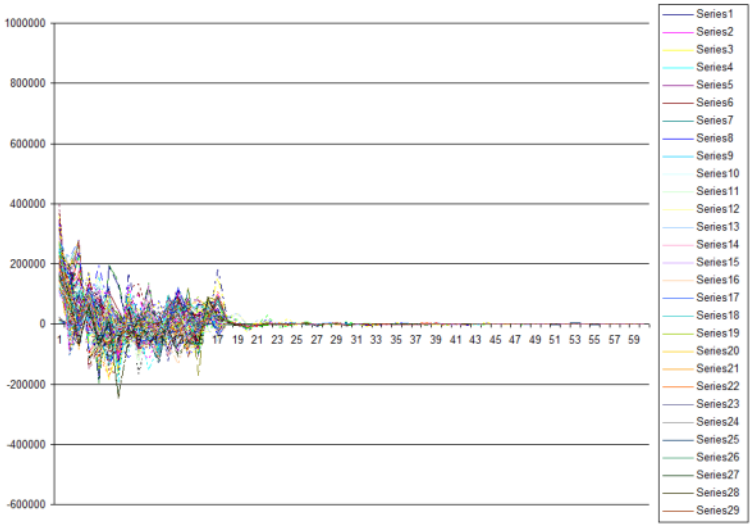


Figure 6.35: PCA of FFT Data, Used for kNN Network Selection or Neural Network Training

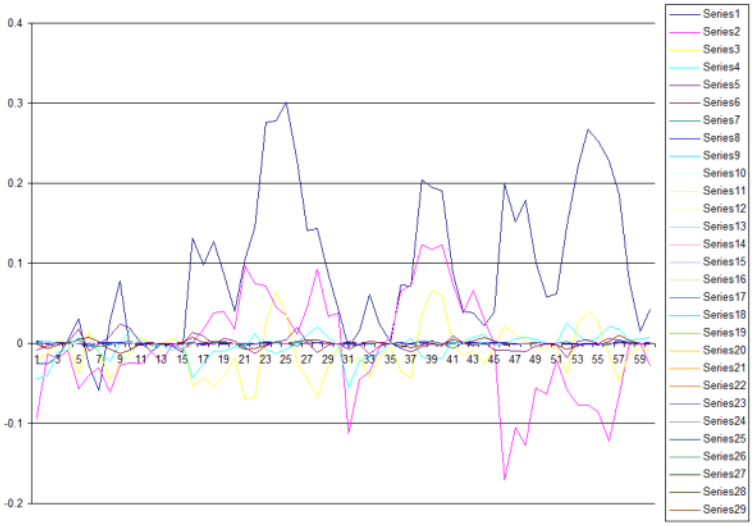


Figure 6.36: Weighted Feature Vector Indices for Use in Nearest Neighbor Selection of Pre-Trained Network

If the user chooses to conduct a training session, the program steps through recording sessions for each activity (Figure 6.33) while storing FFT (Figure 6.34) and EMG (Figure 6.37) or joint angle data in memory. After the baseline is completed, the mean and variance of each channel is calculated and a principle component analysis is performed on the FFT data to generate the feature vector. Once this function is complete, the sigmoid activation neural network training session begins with the option to learn

network weights with PCA on (Figure 6.38) or off (Figure 6.39), which will also change this setting in the control system. Once the neural network has sufficiently converged on a solution, the option to save the neural network to the database is available. This will name the network and feature vector based on the current time, which will be displayed when the network is loaded in future sessions. If the neural network is saved, the feature vector, PCA mode, and weightings are saved to their appropriate database locations in file and the file name is added to the database list. After this function is complete, the real-time control menu is loaded.

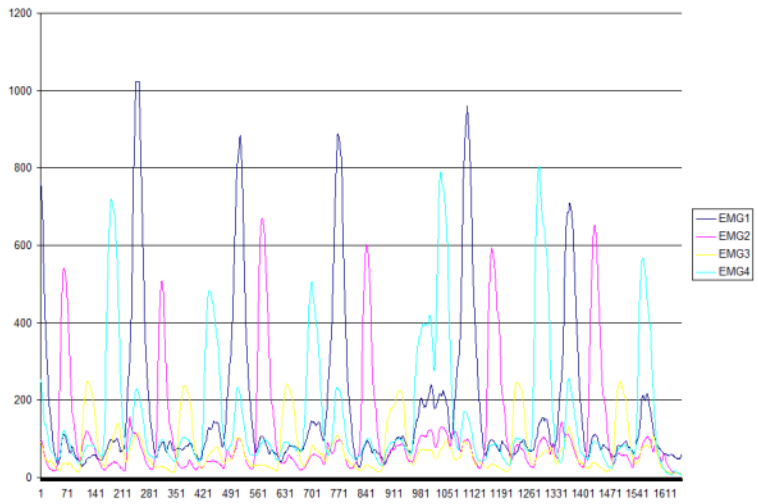


Figure 6.37: EMG Data Collected During Training Sessions

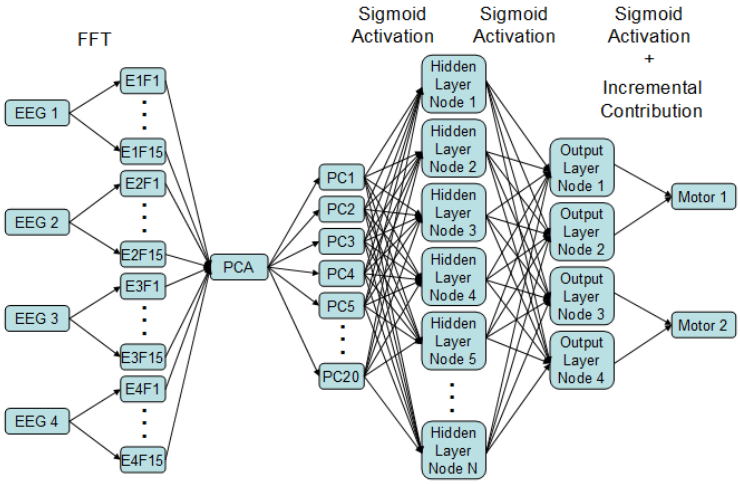


Figure 6.38: Sigmoid Activation Neural Network with PCA Rotation of FFT Data

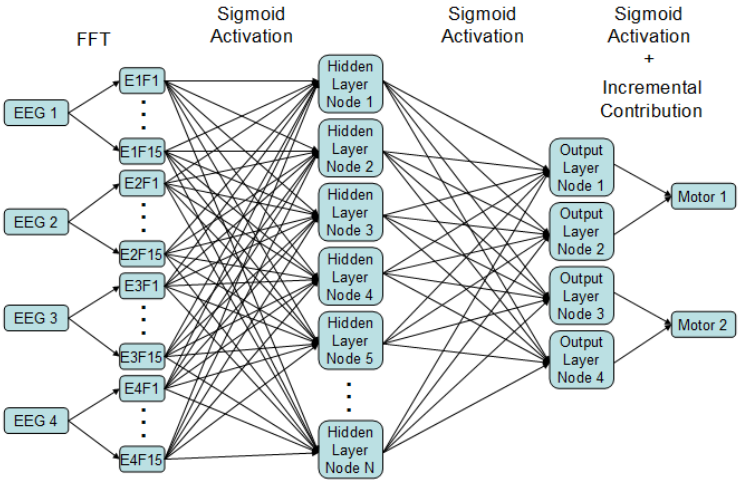


Figure 6.39: Sigmoid Activation Neural Network using FFT Data as Inputs

If the user chooses to record data, the options to either record raw data (Raw EEG: Figure 6.33, Raw EMG: Figure 6.37, and Raw NIRS: Figure 6.40) or pre-processed information (EEG FFT: Figure 6.34, EMG Estimated Joint Position: Figure 6.41, and Blood Oxygenation and Blood Volume Information: Figure 6.42) appear. In either case, the program begins collecting data and writing the selected data type to file until the user presses any key. Exiting the function Record Data returns the user to the start-up menu.

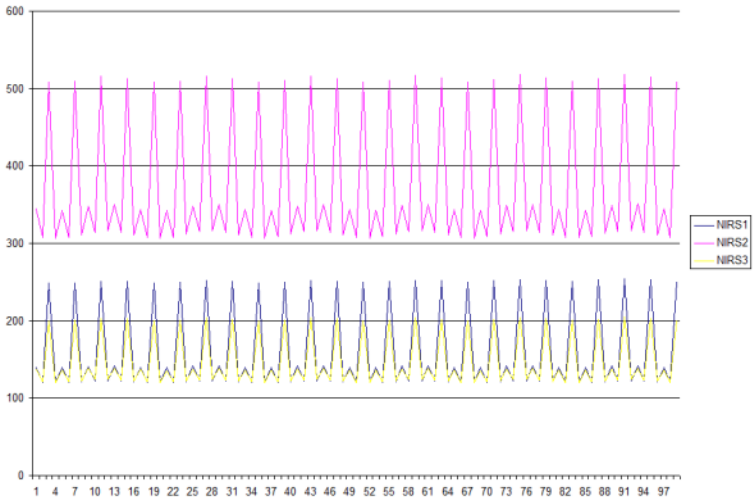


Figure 6.40: Raw NIRS Data: Sensor Data Not Yet Sorted by Active Light Source

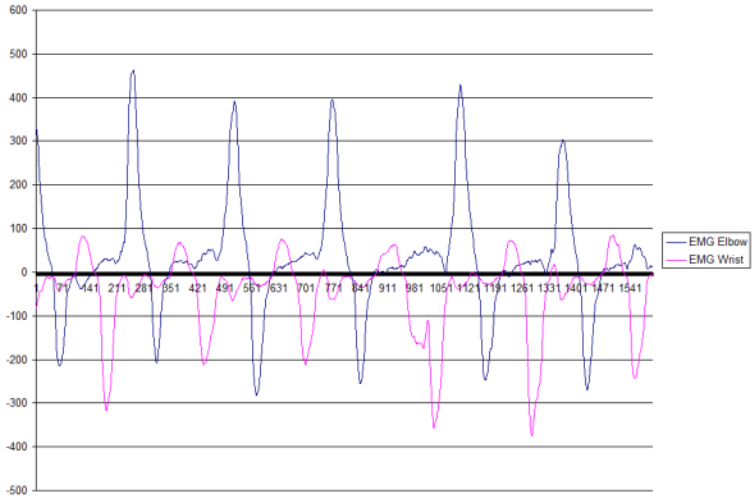


Figure 6.41: EMG Estimated Joint Flexion

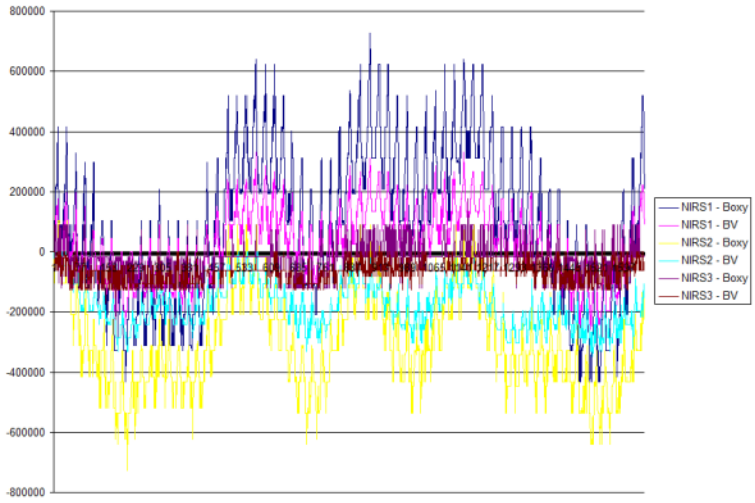


Figure 6.42: NIRS Data, Processed into Blood Oxygenation and Blood Volume

Once a network is either selected from the database or trained, the real-time control menu becomes available. This menu loads the options to perform Standard Control, Mirror Control, or Real-Time Adaptation. These options function very similarly to the real-time control system used in the fuzzy neighbors program, but utilize the sigmoid activation neural network (Figure 6.38 or 6.39) to predict outputs instead of the fuzzy-neighbors algorithm.

Standard control collects data from the headset every control cycle, unpacks the data based on the number of samples collected, performs a Fast Fourier Transform, if in PCA mode the FFT is rotated into the PCA space, the inputs are given to the neural network, output values are generated (Figure 6.43), and a proportional gain controller with decay (Equation 6.1) is used to calculate the value sent to the arm control motors based on the current neural network output (Figure 6.44). Mirror control simply mirrors the headset data channels prior to analysis.

$$M_i = (1 - D) * (M_{i-1} - G * (O_{j+1} - O_j)) \qquad 6.1$$

Where M_i is the motor output value, D is the rate of decay, which is related to the gain G by Equation 6.2, M_{i-1} is the previous motor position, and O_j and O_{j+1} are opposing muscle outputs of the neural network.

$$G = L_i * D \qquad 6.2$$

Where L_i is the half of the range available for M_i.

Using this controller results in the maximum output never exceeding the motor position limits of $[-L_i, L_i]$, as a result of the ranges of O_j and O_{j+1} being bounded [0, 1] from the sigmoid activation function, and the test arm has a disposition to return to the neutral position. Using the servo motor range [0, 1023], L_i is set to 512 and the motor outputs are re-centered on this range prior to output to the motors.

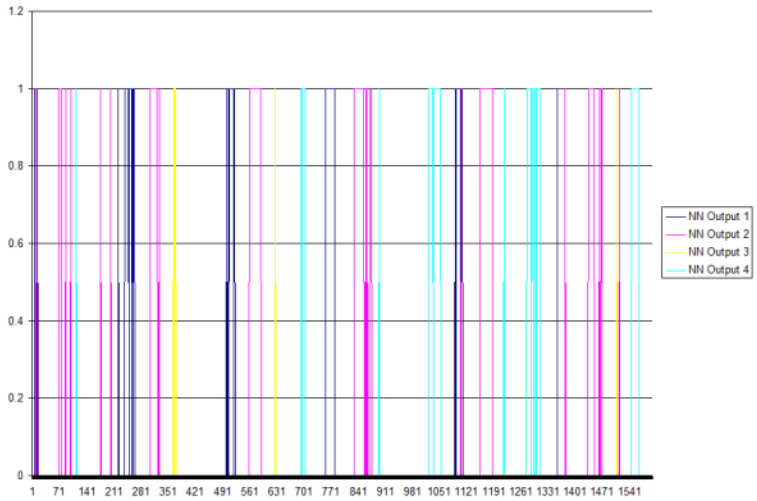


Figure 6.43: Raw Outputs of Neural Network

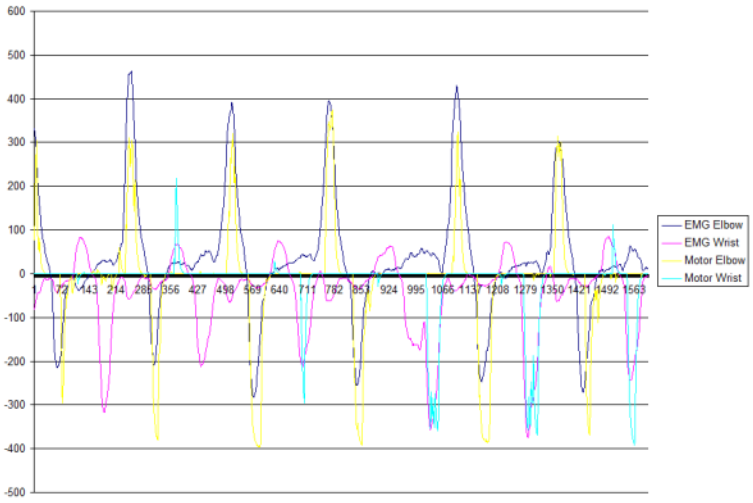


Figure 6.44: Comparison of Neural Network Output with EMG-Joint Flexion

Real-time adaptation functions the same as standard control with the addition of NIRS analysis and periodic neural network weight updates. Each control cycle the system calculates the blood oxygenation and blood volume at each NIRS sensor (Figure 6.42). If the blood volume in a region becomes elevated for more than 5 seconds, the system begins comparing the NIRS activation with the neural network output. If the two systems agree on the activity being performed, the neural network uses the current input and output data to perform one pass of adaptation in order to reinforce the positive outcome. If the two systems disagree on the activity being performed, the neural network uses the current inputs and zero values on the outputs to perform one pass of adaptation to mitigate the error that is occurring. When the third generation headset is available for testing, the capabilities of this function are expected to greatly improve and extensive healthy subject testing will be conducted. When real-time adaptation is terminated by the user, the option to save the current network to the database is once again presented.

At this time the software does not have an interface to view pre-trained networks in the database and make edits or deletions. Users may only make additions. Edits and deletions may however be performed manually by individuals familiar with the program and database structure.

The neural network performance using this system could still see some improvement, but some of the test sets of data already perform well using the initial settings. Training a neural network without PCA generally yields the most consistent results (Figures 6.45 – 6.52), but PCA is useful in some situations where a normal input neural network has difficulty training to the data set (Figures 6.53 and 6.54). The inconsistency between convergence accuracy and control performance appears to be related to rapid classification changes in the PCA neural network output. Figures 6.45 and 6.47 represent sufficiently accurate control to proceed with healthy subject testing. Figure 6.45 indicates that forty-eight activities occurred, including resting periods between joint motions, and only five actions resulted in an incorrect output, producing an accuracy of 89.6% in use. The root mean square error of the elbow position is approximately 11.87% and 15.42% for the wrist joint.

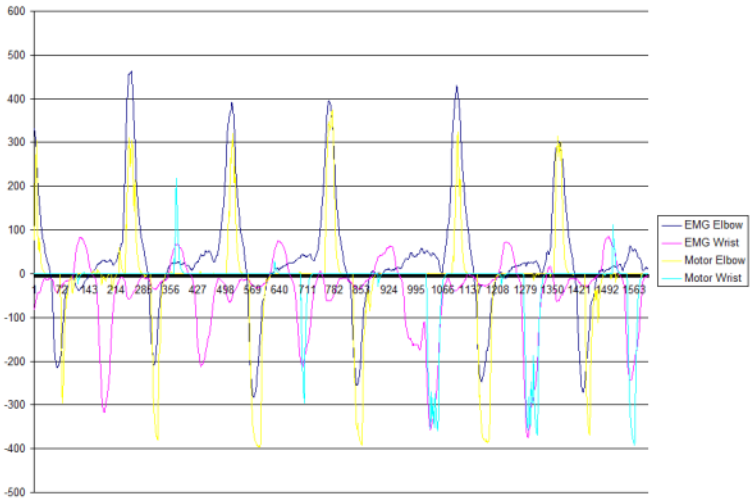


Figure 6.45: Data Set 1655, PCA Off, 94.625% Accuracy at 1000 Epochs

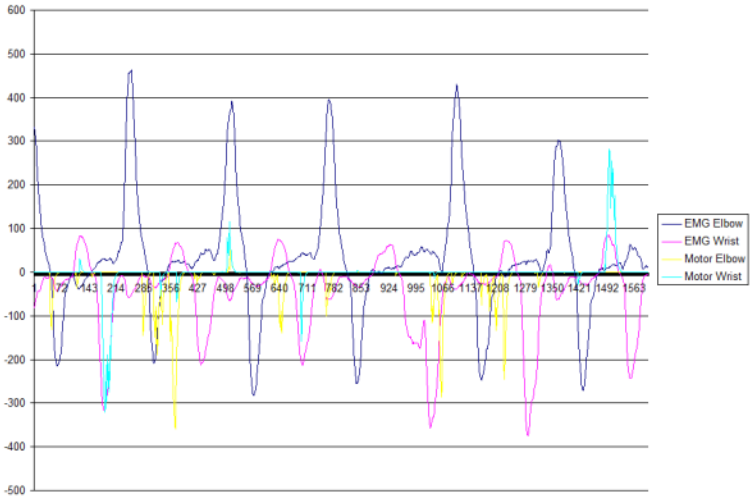


Figure 6.46: Data Set 1655, PCA On, 96.875% Accuracy at 1000 Epochs

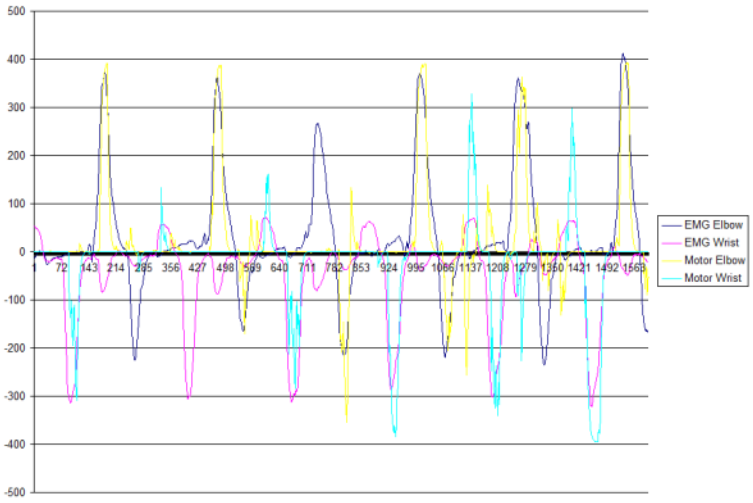


Figure 6.47: Data Set 4029 PCA Off, 95.0625% Accuracy at 1000 Epochs

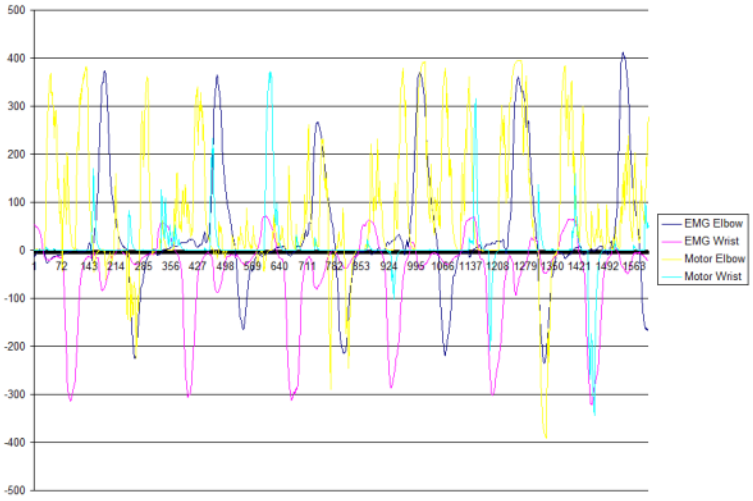


Figure 6.48: Data Set 4029 PCA On, 90.6875% Accuracy at 1000 Epochs

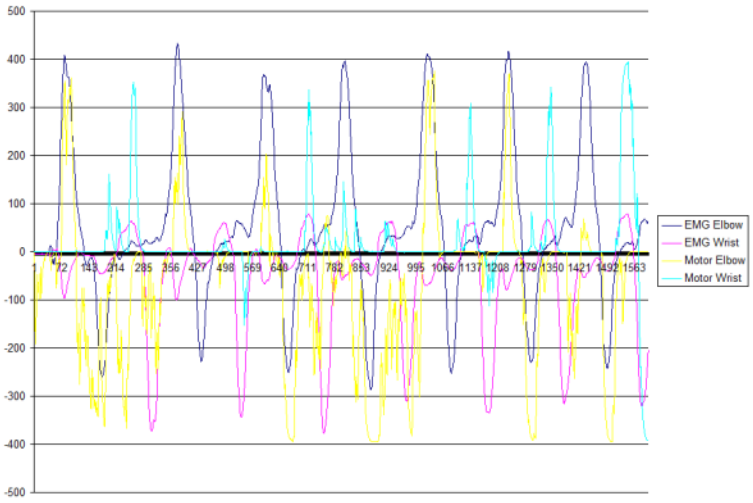


Figure 6.49: Data Set 4258 PCA Off, 88.75% Accuracy at 1000 Epochs

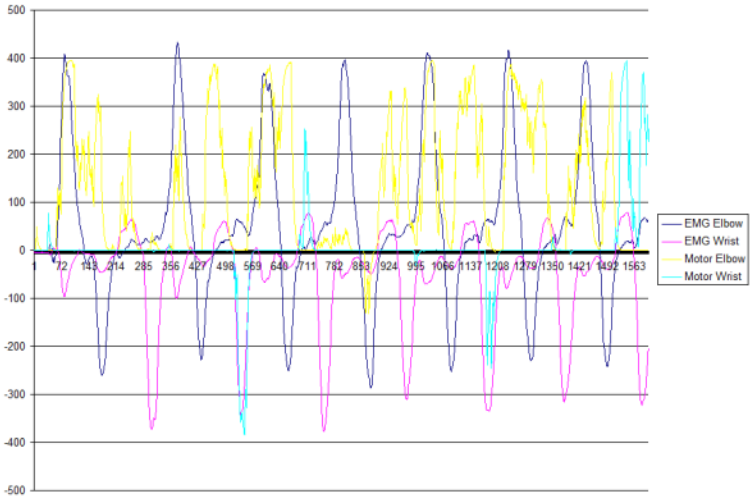


Figure 6.50: Data Set 4258 PCA On, 81.75% Accuracy at 1000 Epochs

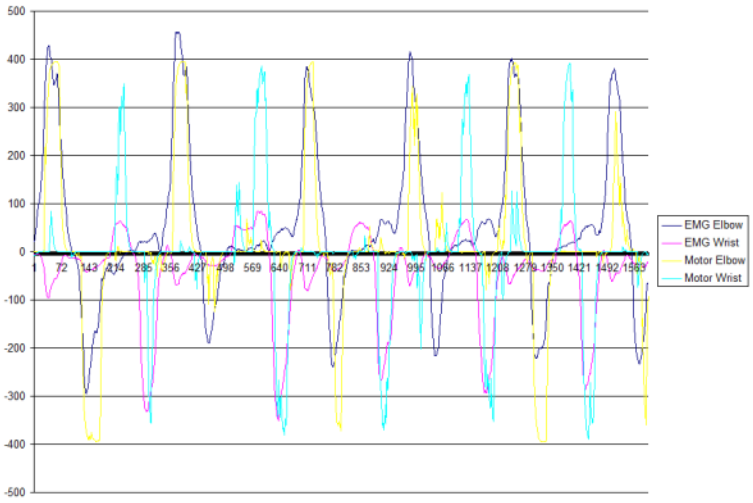


Figure 6.51: Data Set 5035 PCA Off, 95.125% Accuracy at 1000 Epochs

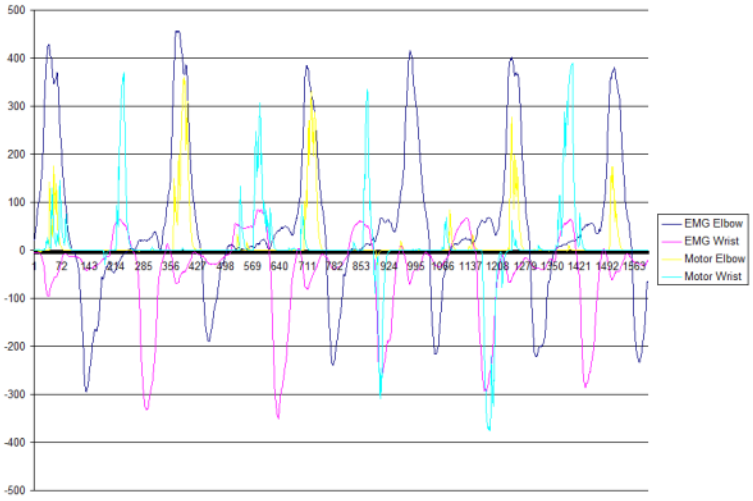


Figure 6.52: Data Set 5035 PCA On, 95.1875% Accuracy at 1000 Epochs

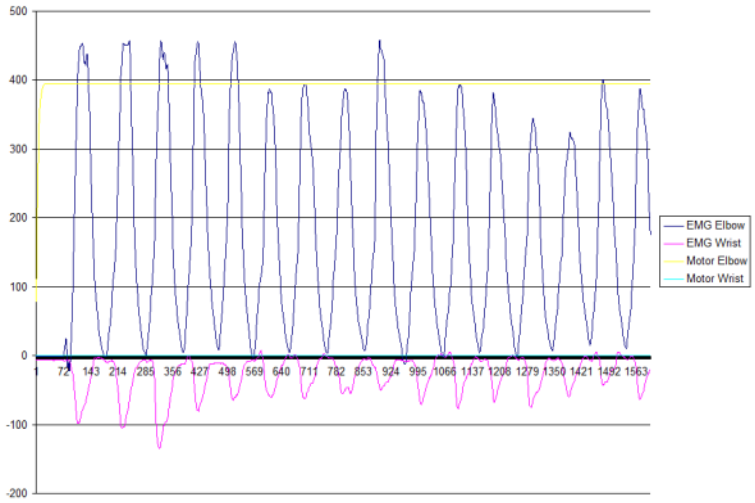


Figure 6.53: Data Set 6570 PCA Off, 31.3125% Accuracy at 1000 Epochs

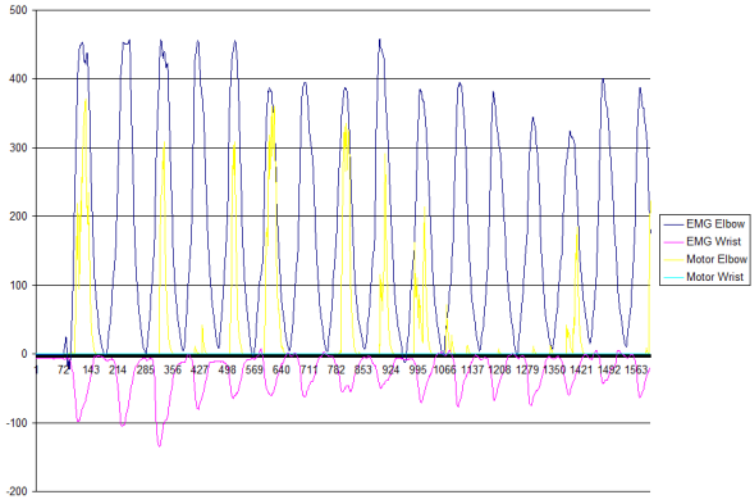


Figure 6.54: Data Set 6570 PCA On, 81.75% Accuracy at 1000 Epochs

Inconsistencies resulting from training methods have lead to the development of a new training routine that allocates equal amounts of sample data for each activity, including resting. We believe the combination of more controlled data collection and the improved sensors capabilities of the third generation headset will be sufficient to provide consistent control of the artificial limb. The figures for data set 4258 (Figures 6.49 and

6.50) in particular demonstrate how much impact input noise has on the output of the control system.

CHAPTER SEVEN: SUMMARY AND CONCLUSIONS

Summary

This research focuses on the design and development of a non-invasive system for monitoring brain activity for the purpose of control of an artificial limb. Through proof of concept preliminary data collection and analysis, it was demonstrated that a combination of EEG and NIRS could be used to discern brain activity and correlate that activity to EMG data collected simultaneously. While simple control methods were only able to produce basic actuation, machine learning techniques, particularly neural networks, have proven to be capable of identifying multiple output activities based on the input data.

The development of an improved brain monitoring device as well as a data acquisition and control system have been explored, resulting in a design that is ready to see extensive healthy subject testing. Pending positive results from healthy subject use, the system should be ready to be used in clinical trials for limb replacements as well as phantom limb pain therapy.

Transfer Function

From these results it can be concluded that using the difference and average of corresponding EEG channels is a more effective method for data analysis. While the estimation of EMG activity is similar between methods, the noise cancellation is much improved by this technique. NIRS is being used in these results to attenuate or amplify

the variable gain used for motor control. However, in the limited duration of these results, the cumulative effects of the adaptation algorithm are minor. Despite the improvements resulting from these modifications, using a transfer function still produces very poor results when performing more complex actions. Bicep actuation occurs during nearly any movement, suggesting that the correlation between inputs and outputs can not be clearly defined by simple relationships.

Based on the results from testing this simple control algorithm, it was determined that a more robust control scheme should be possible by increasing the number of sensors and investigating more advanced training and adaptation algorithms. For this purpose a new sensor system was constructed and has undergone initial testing to verify the performance of the equipment.

Case Matching Algorithm

The case matching algorithm produced very poor results and was determined to be insufficient for this application.

Fuzzy-Nearest Neighbors

While the initial results of this system show promise, the probable performance in implementation indicates poor results. For this reason the fuzzy-neighbors algorithm has been removed as a potential solution and a k-nearest neighbors algorithm was investigated.

k-Nearest Neighbor

While the results of this algorithm are better than the fuzzy-neighbors program, the issues regarding identifying data not recorded in the same session as the member sets is a serious concern for this application. While it may be theoretically possible to include enough sets of data in the known neighbors, performing the necessary calculations could become computationally challenging considering the results in this study already used 19,386 (10-Fold) and 17,232 (5-Fold) sets. For this reason an artificial neural network was investigated for the purpose of loading more of the computation into the control algorithm preparation so that fewer calculations have to be performed in real-time. This would hopefully also provide better regression than the k-nearest neighbors technique.

Linear Activation Neural Network

The fact that this method could not converge on a solution indicates that a linear activation function is not appropriate for our use. The activation function must be differentiable in order to perform error gradient descent to arrive at a valid solution. If an activation function is to be used, the input and output data would need to be normalized. If data manipulation would already be removing our control system from the [0-1023] output resolution that we had been operating with, then additional techniques should also be explored to improve the data analysis prior to being used in the control system. For this reason a Principle Component Analysis (PCA) algorithm was investigated in order to prepare for using a more common artificial neural network.

Principle Component Analysis

PCA is clearly a useful technique to reduce this data type into fewer dimensions. This will allow a neural network to have fewer input nodes and hopefully offset enough computation to merit rotating the FFT data into the PCA coordinate system at each sample. The benefits of PCA on neural network performance were explored using the nonlinear network, with the intention of simplifying the input data complexity enough to recognize activities similar to the training set.

Additionally, it was concluded that PCA is a useful technique for recognizing similarities between brain activity from different individuals. Once sample data has been collected, it may be possible to compare the feature vector of a new subject with a database of existing feature vectors and by using the k-nearest neighbor algorithm, with k set to one, a similar brain mapping can be pulled from a database of pre-trained neural networks linked to the feature vectors. This would make training a neural network for a new subject much easier in cases where output data cannot be collected.

Sigmoid Activation Neural Network

Based on the results using this method, it can be concluded that a neural network as a regression algorithm does not perform satisfactorily for this application. However, using a neural network as a classifier provides sufficient results. This indicates that a control system that is designed to actuate incrementally, or based on the confidence level of the classification, would be sufficient to meet the requirements of this project. For this

reason, a sigmoid activation neural network, trained using real-time and/or training session data, will be used in the implemented control system for this project.

Conclusions

The highest accuracy achieved using each method (excluding small sample sizes) is presented in Table 7.1.

Table 7.1: Best Performance for Each Method

Method	Independent Data Set Test Accuracy	K-Fold Test Accuracy	K-Fold Root Mean Square Error
Fuzzy Neighbors	0.067%		
k-Nearest Neighbors	41%	93.22%	~31.69%
Linear Regression Neural Network	24.3%		
Non-Linear Regression Neural Network	23.70%		
PCA - Non-Linear Regression Neural Network	20.70%		
Non-Linear Classifier Neural Network	60.60%	90.27%	21.68%
PCA - Non-Linear Classifier Neural Network	66.40%	86.68%	26.39%

While the k-nearest neighbors algorithm's K-Fold results indicate it is the best performing technique, the drop in accuracy against non-similar data indicates that it is not useful for our application due to real-world variation in input data. Artificial neural networks using a non-linear activation function perform sufficiently well and don't suffer from as much sensitivity to changing sensor conditions. The option to use PCA will be included in future designs for its improved performance in very poor sensor conditions, but the highest accuracies are likely to result from proper equipment use and selection of an appropriate pre-trained neural network.

Verification That Design Specifications Were Met

While testing against a data set from a separate training session has fallen under the requirements of this study, the potential to reach the stated goals has been

demonstrated through the use of a non-linear neural network and k-folding to provide similar test data. Testing on unique, but similar data, has been demonstrated to achieve an accuracy of 90.27% when using a viably large data set, which exceeds the requirement of 80% for output classification. While regression using this method provided poor results, using the neural network as a classifier combined with actuation of the artificial limb based on the confidence of the neural network output resulted in an accuracy of 89.6% in the implementation test. The root mean square error of the elbow position is approximately 11.87% and 15.42% for the wrist joint, which is within the 20% margin stated in the requirements.

Advancements to the Field of Study

- This research has resulted in a novel solution to using NIRS and EEG data in a non-invasive brain-computer interface.
- This research has resulted in advances to the field of artificial limb control, which may have a dramatic impact on the prosthesis market as most research in this field is focused on invasive technologies in order to provide the necessary level of control.
- At the level of accuracy provided by the second generation headset, this research has reached a level of performance appropriate for use in clinical treatment of phantom pain, where the output of the artificial limb is second to the ability to provide a connection between brain activity and physical outcome.
- The results from testing different numbers of hidden nodes, while maintaining other aspects of the neural network, may confirm research indicating that over-fitting a neural network does not have a negative impact on the networks ability to perform classification.
- From this research it can be concluded that electrical brain activity and musculoskeletal motion do not share a simple, linear relationship.

- This research has demonstrated that neural networks are a feasible solution to mapping brain activity, which may be used in other applications such as forensics.

Recommendations for Future Work

The third generation NIRS/EEG headset will produce more reliable data through improvements to reference voltage design, sensor quantity, sensor positioning, and wear-ability. It is our hope that through these improvements, the variation of sensor data between recording sessions will be dramatically improved. With additional NIRS channels over relevant motor cortex regions, it should also be possible to improve the algorithm for online adaptation through the ability to discern limb section activation and not only side-of-body activations. We believe that the techniques developed in the course of this research should be sufficient in combination with the new sensor headset, to enter clinical trials. Healthy subject testing will commence shortly after the headset is complete in order to verify this hypothesis.

It may also be beneficial to investigate the use of state vector machines and wavelet transforms. These techniques could provide the capacity to account for changing sensor conditions and provide a more robust solution than the neural network. Additional data analysis prior to entry into the control system may also be beneficial and will be examined following the next round of healthy subject testing. It has been indicated in literature that the frequencies 30-100 Hz may also be useful for distinguishing motor control and should be investigated. Another approach to improving distinction between activities is to switch the learning output to be provided through joint angle and velocity measurement. By deviating from the biological output of EMG, the system will be able to

track a wider variety of activities with greater precision, but will no longer be a brain-to-muscle mapping as the project originally intended. An exoskeleton to track joint angles is currently being developed.

REFERENCES

Aasted, Christopher M. “Direct Brain Control of Prosthesis.” Masters Thesis, University of Denver, 2008.

Buxton, Richard B., Uludag, Kamil, Dubowitz, David J., and Liu, Thomas T. “Modeling the hemodynamic response to brain activation.” *NeuroImage* 23 (2004): 220-33.

Carey, J. *Brain Facts*. Society for Neuroscience, 2008. Available from http://www.sfn.org/skins/main/pdf/brainfacts/2008/neuron.pdf; Internet; accessed 20 April 2011.

Caruana, Rich, and Niculescu-Mizil, Alexandru. “An Empirical Comparison of Supervised Learning Algorithms.” Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.

Castillo, Eduardo et al. “Integrating sensory and motor mapping in a comprehensive MEG protocol, Clinical validity and replicability.” *NeuroImage Uncorrected Proof* (2003): 11 pages.

Center for Disease Control and Prevention. “Arthritis – At A Glance 2009.” Available from http://www.cdc.gov/nccdphp/publications/aag/pdf/arthritis.pdf; Internet; accessed 2 October 2009.

Collura, T. F. “Initial Results with 2-channel EEG and Sum/Difference Mode using BrainScape.” Available from http://www.brainmaster.com/tfc/index_files/Publications/2chanbrainscapes.pdf; Internet; accessed 16 May 2005.

Darlot, C., Eskiizmirliler, S., Forestier, N., and Tondu, B. “A model of the cerebellar pathways applied to the control of a single-joint robot arm actuated by McKibben artificial muscles.” *Biological Cybernetics* 86 (2002): 379-394.

Dasarathy, Belur. “Minimal Consistent Set (MCS) Identification for Optimal Nearest Neighbor Decision Systems Design.” *IEEE Transactions on Systems, Man, and Cybernetics* 24 (March 1994): 511-517.

de Castro, L. N. and Von Zuben, F. J. “An Immunological Approach to Initialize Feedforward Neural Network Weights.” Proc. of ICANNGA (Int. Conf. on Artificial Neural Networks and Genetic Algorithms). Prague/C.R., 22-25 April 2001.

Deecke, Luder, Beisteiner, Roland, and Lang, Wilfried. “Human voluntary movement physiology as studied by DC-EEG, MEG, SPECT and FMRI” *Electroencephalography and Clinical Neurophysiology* 47 (1996): 295-311.

Defense Sciences Office. “Revolutionizing Prosthetics.” Available from http://www.darpa.mil/dso/thrusts/bio/restbio_tech/revprost/; Internet; accessed 14 March 2010.

Defensetech.org. “Replacement Arm, Good As New.” Available from http://www.defensetech.org/archives/001478.html; Internet; accessed 11 April 2005.

Devor, Anna, Ulbert, Istvan, Dunn, Andrew K., Narayana, Suresh N., Jones, Stephanie R., Andermann, Mark L., Boas, David A., and Dale, Anders M. “Coupling of the cortical hemodynamic response to cortical and thalamic neuronal activity.” Proceedings of the National Academy of Sciences. March 8, 2005.

Donchin, Opher, Francis, Joseph T., and Shadmehr, Reza. “Quantifying Generalization from Trial-by-Trial Behavior of Adaptive Systems that Learn with Basis Functions: Theory and Experiments in Human Motor Control.” *The Journal of Neuroscience* 23 (2003): 9032-9045.

Duan, Kai-Bo and Keerthi, S. Sathiya. "Which Is the Best Multiclass SVM Method? An Empirical Study." Proceedings of the Sixth International Workshop on Multiple Classifier Systems. 2005.

Emotiv. “Emotiv EPOC™.” Available from http://emotiv.com/corporate/2_0/2_2.htm; Internet; accessed 2008.

Englehart, Kevin; Hudgins, Bernard. “A Robust, Real-Time Control Scheme for Multifunction Myoelectric Control.” *IEEE Transactions on Biomedical Engineering* 50 (2003): 848-54.

Eskiizmirliler, S., Tondu, B., and Darlot, C. “Motor control of a limb segment actuated by artificial muscles.” Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 1, 2001: 865-8.

Gale Group, The. “Institute of Physical & Chemical Research’s artificial hand measures voltage nerves send to muscles from brain; automatically controls ultrasonic motor movement.” *Nikkan Kogyo Shimbun* (1995).

Guger Technologies. “g.EEGcap.” Available from http://www.gtec.at/products/g.Accessories/gEEGcap.htm; Internet; accessed 2008.

Guoqiang, Yu et al. “Hemodynamic measurements in rat brain and human muscle using diffuse near-infrared absorption and correlation spectroscopies.” Proceeding of the SPIE – The International Society for Optical Engineering 4955 (2003): 164-74.

Heaton, Jeff. *Introduction to Neural Networks for Java, 2nd Edition.* Chesterfield, MO: Heaton Research, Inc, 2008.

Heeger, D.J. and Ress, D. “What Does fMRI Tell Us About Neuronal Activity?” *Nature Reviews: Neuroscience* 3 (2002): 142-151.

Hitachi-Medical. “Principle of Optical Topography System - System Configuration.” Available from http://www.hitachi-medical.co.jp/info/opt-e/genri-4.html; Internet; accessed 2008.

Ho, Tin. "The Random Subspace Method for Constructing Decision Forests". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (8) (1998): 832–844.

Hoshino, Tatsuya et al. “Application of multichannel near-infrared spectroscopic topography to physiological monitoring of the cortex during cortical mapping: technical case report.” *Surgical Neurology* 64 (2005): 272-275.

Huettel, Scott A. and McCarthy, Gregory. “Evidence for a Refractory Perior in the Hemodynamic Response to Visual Stimuli as Measured by MRI.” *NeuroImage* 11 (2000): pages 547-53.

Huettel, Scott A., Singerman, Jeffrey D., and McCarthy, Gregory. “The Effects of Aging upon the Hemodynamic Response Measured by Funtional MRI.” *NeuroImage* 13 (2001): 161-175.

Immrama Institute. “The International 10-20 System of Electrode Placement.” Available from http://www.immrama.org/eeg/electrode.html; accessed 14 March 2010.

Kram, Barbara. “Revolutionizing Prosthetics 2009 Team Delivers First DARPA Limb Prototype.” Available from http://www.dotmed.com/news/story/3875; Internet; accessed 2008.

Lawrence, Steve; Giles, C. Lee; Tsoi, Ah Chung. “Lessons in Neural Network Training: Overfitting May be Harder than Expected.” Proceedings of the Fourteenth National Conference on Artificial Intelligence, Menlo Park, California, 1997.

Muller, H. -P. et al. “Magnetoencephalographic and Functional Magnetic Resonance Imaging in a Single Software Environment.” *IEEE Engineering in Medicine and Biology* May/June (2005): 109-16.

Nemes, S et al. “Bias in odds ratios by logistic regression modeling and sample size.” *BMC Medical Research Methodology* 9:56 (July 2009).

Pomerleau, Dean. *Knowledge-based training of artificial neural networks for autonomous robot driving.* Edited by J. H. Connell and S.Mahadevan. *Robot Learning.* New York: Kluwer Academic Publishers, 1993.

Purves, Dale, Fitzpatrick et al, eds. "The Premotor Cortex," *Neuro Science* 2nd Edition. Sunderland, MA: Sinauer Associates, 2001.

Saygin, Ayse Pinar, Wilson, Stephen M., Hagler, Donald J. Jr, Bates, Elizabeth, and Sereno, Martin I. "Point-Light Biological Motion Perception Activates Human Premotor Cortex." *The Journal of Neuroscience* 24 (27) (2004): 6181-8.

Segal, Mark. "Machine Learning Benchmarks and Random Forest Regression." *Center for Bioinformatics & Molecular Biostatistics, UC San Francisco*. Available from http://escholarship.org/uc/item/35x3v9t4; Internet; accessed 3 May 2011.

Webb, Andrew. *Introduction to Biomedical Imaging*. New York: Wiley, 2003.

Vapnik, Vladimir. *The Nature of Statistical Learning Theory.* Verlag: Springer, 1995.

Villringer, A. and Chance, B. "Non-invasive optical spectroscopy and imaging of human brain function." *Trends in Neuroscience* 20 (10) (1997): 435-442.

Virtualworldlets.net. "Dean Kamen's Robotic "Luke" Arm Demonstrated." Available from http://www.virtualworldlets.net/Archive/IndividualNews.php?News=2971; Internet; accessed 6 June 2008.

Winter, David. *Biomechanics and Motor Control of Human Movement*. Hoboken, New Jersey: Wiley, 2005.

Yacoub, E. and Hu, X. "Detection of the early decrease in fMRI signal in the motor area." *Magn. Reson. Med*. 45 (2) (2001): 184-190.

Zhang, Harry. "The Optimality of Naive Bayes". FLAIRS2004 conference.

Ziegler-Graham, Kathryn et al. "Estimating the Prevalence of Limb Loss in the United States - 2005 to 2050." *Archives of Physical Medicine and Rehabilitation* 89 (2008): 422-429.

APPENDIX A: DATA SHEETS

A.1: NIRS System Specifications Used for Proof of Concept Data Collection

6. APPENDIX – SPECIFICATIONS

1. Physical Specifications

1.1. Dimensions of Probe: 18x6x0.8cm

1.2. Probe Materials: Foam & Wire

1.3. Number of detectors: 10

1.4. Number of light sources: 4

1.5. Light-Source Separation: 2.5 cm

2. Other Physical Specifications

2.1. Dimensions of Imager Electronics Unit: 25.5x24.5x10cm

2.2. PC Interface card Specs: PCMCIA National Instruments DAQ-1200

3. Functional Specifications

3.1. Time to stabilize: 15 seconds max

3.2. Time to balance gain: 10 seconds max

3.3. Dark current correction: 0 seconds min

4. Sample Rate Specs

4.1. Sample rate = 60,000 samples/second or A/D Conversion every 16.7 microseconds

4.2. 250 samples per LED wavelength per detector = 4.2 milliseconds

4.3. 2 wavelengths per detector = 12.5 milliseconds

4.4. For 16 channels = 200 milliseconds for one whole sample

4.5. Timer interrupt in the software is set to 330ms (full image rate = 3 Hz)

5. LED Specs (Epitex L6X730/6X850)

5.1. 2 wavelengths: 730, 850nm (+-15nm)

5.2. Max continuous current/relative power at each wavelength

5.2.1. 730: 75ma/90%

5.2.2. 850: 100ma/100%

6. Opto101 Photodiode Specs:

6.1. Active area: 2.3x2.3mm

6.2. Unit to unit detection variation +-5%

6.3. Non-linearity: .01% of full signal

6.4. Bandwidth 14kHz

6.5. Voltage output Vs. Temperature: 100ppm/C

6.6. Dark Current: 7.5mv

A.2: EEG System Used for Proof of Concept Data Collection

Atlantis I Clinical System

EEG/Neurofeedback Systems

Item #: AT1-A1CLIN
Favorites
Your Price: $2,695.00
1 Buy

ATLANTIS I is a 4x4 design (with four channels of EEG and four channels of AUX signals) ideal for additional biofeedback modalities. Includes continuous real-time impedance checking and recording; Total Immersion with photic, vibratory and auditory feedback. Evolving 24 bit USB technology.

Flextrode Package 2

Electrodes & Headgear

Item #: 621-022
Favorites
Your Price: $240.00
1 Buy

FLEXTRODE Package 2

HearGear product which eliminates the need for paste and goo! Interfaces with other neurofeedback products on the market.

Flextrode offers an easy, quick and clean approach to neurofeedback training.

(Patent #6,574,513)
Consists of a hollow, screwlike plastic retainer containing a felt wrapper that encloses a standard gold cup electrode. They are used with a retaining ring that can be inserted in a headband, hat, or other holder. This provides a simple, clean, self-abrading and self-cleaning, paste-free and gel-free contact. Used for EEG, EMG, EKG and related body potential recording. Ideal in cases where it is difficult to get a good connection, especially through hair. Electrode lead color our choice. Easy to apply and remove. Leaves hair clean without a trace of goo. A must in keeping teenagers and adults happy to come back.

Includes 3 FlexTrode Assemblies, Headband and Cross-strip, 3 electrodes, plus Flextrode electrolyte solution and bottle. plus two earclips, NuPrep Gel, 10/20 Paste. Eliminates need for 10-20 paste on head. Keeps hair clean . SAF Non returnable for hygenic reasonsSAF (621-022):

$235.00 ($200.00)

A.3: EMG Sensors Used for Feedback Data Collection

LifeMatters
TOOLS FOR STRESSLESS LIVING

MyoTrac Portable Muscle Monitor
Train Your Body to Mind
FREE Unlimited support with a Biofeedback Therapist

Market Price $499.95
Our Price $457.97
Price Guarantee
Add to Cart

In Stock
Ships within 1 business day
U.S. Shipping Cost - $8.00
Shipping Method - UPS
International Ship Cost- see at checkout
International Ship Method - Global Exp
General policies and guarantee

Specifications

MyoTrac SA4000P/SA4001P Hardware Specifications

Size (approx.)	61mm x 112mm x 25mm (2.4" x 4.4" x 1")
Weight (approx.)	73g (2.5oz)
Input Impedance	1,000,000MΩ in parallel with 10pF
Signal Input Range x1	0 – 20μV
Signal Input Range x10	0 – 200μV
Signal Input Range x100	0 – 2000μV
CMRR 20 – 500 Hz (approx.)	130dB
CMRR @ 60 Hz (approx.)	180dB
Channel Bandwidth (Wide)	20Hz ±5Hz – 500Hz ±50Hz
Channel Bandwidth (Narrow)	100Hz ±10Hz – 200Hz ±20Hz
Signal Output Range	0V – 2V
Supply Voltage	7.4V – 9.6V
Current Consumption	10mA – 3mA /+5mA
Battery Life (Alkaline)	40 Hours minimum
Low Battery Warning	7.4V
Accuracy	±5% and ±0.3mV

A.4: LED Light Source Used for Preliminary NIRS Study

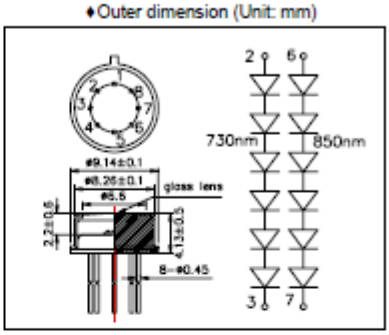
epitex *Opto-Device & Custom LED* Φ8 STEM TYPE LED L6*730/6*850-40Q96-I

Lead (Pb) Free Product RoHS compliant

L6*730/6*850-40Q96-I multi-wavelength LED

L6*730/6*850-40Q96-I consists of12chips of AlGaAs (730nm and 850nm) LED mounted with AlN heat sink pedestal on TO-5 stem and sealed with a flat glass can.

♦Outer dimension (Unit: mm)



♦Specifications

1) Product Name	Multi-wavelength LED Lamp
2) Type No.	L6*730/6*850-40Q96-I
3) Chip	
(1) Chip material	AlGaAs
(2) Peak wavelength	730, 850nm
(3)Chip numbers	Each 6pcs
4) Package	
(1) Stem	TO-5, 8 pins type
(2) Lens	Flat glass can

♦Absolute Maximum Ratings [Ta=25°C]

Item	Symbol	Maximum Rated Value		Unit
		730	850	
Power Dissipation	PD	1100	1280	mW
Forward Current	IF	75	100	mA
Pulse Forward Current	IF	200	500	mA
Reverse Voltage	VR	5		V
Operating Temperature	TOPR	-20 ~ +80		°C
Storage Temperature	TSTG	-30 ~ +100		°C
Soldering Temperature	TSOL	240		°C

‡Soldering condition: Soldering condition must be completed within 3 seconds at 240°C and s allowed in the area apart 3mm from the bottom of the lamp.

◆Electro-Optical Characteristics [Ta=25°C]

Symbol	Wavelength	Condition	Minimum	Typical	Maximum	Unit
VF	730	IF=50mA		11.0	14.0	V
	850			8.5	11.0	
IR		VR=10V			10	uA
PO	730	IF=50mA		50		mW
	850			60		
λP	730	IF=50mA		730		nm
	850			850		
Δλ	730	IF=50mA		30		nm
	850			40		

‡Total Radiated Power is measured by Photodyne#500

EPITEX INC. : 4, Nishiaketa-Cho, Higashi-Kujyou, Minami-Ku, Kyoto, Japan
Tel.: ++81-75-682-2338 Fax: ++81-75-682-2267
e-mail: sales-dep@epitex.com http://www.epitex.com

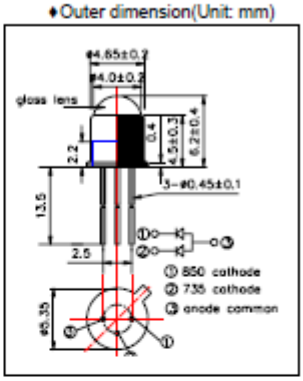
A.5: LED Light Source Used for NIRS

epitex *Opto-Device & Custom LED* Φ5 STEM TYPE LED L735/850-40D32

L735/850-40D32 multi-wavelength LED

L735/850-40D32 consists of an AlGaAs'(735, 850nm), LED mounted on TO-18 stem with a glass ball lens. On forward bias it emits a spectral band of radiation, which peaks at 735nm and 850nm without sub-peak.

♦Outer dimension(Unit: mm)



♦Specifications

1) Product Name	Multi-wavelength LED Lamp
2) Type No.	L735/850-40D32
3) Chip	
(1) Chip material	AlGaAs,
(2) Chip dimension	400um
(3) Peak wavelength	735nm, 850nm
4) Package	
(1) Stem	TO-18 3pins type
(2) Lens	Φ5mm glass ball lens

♦Absolute Maximum Ratings [Ta=25°C]

Item	Symbol	Maximum Rated Value		Unit
		735	850	
Power Dissipation	P_D	180	160	mW
Forward Current	I_F	75	100	mA
Pulse Forward Current	I_F	200	500	mA
Reverse Voltage	V_R	5		V
Operating Temperature	T_{OPR}	-20 ~ +80		°C
Storage Temperature	T_{STG}	-30 ~ +100		°C
Soldering Temperature	T_{SOL}	240		°C

‡Soldering condition: Soldering condition must be completed within 3 seconds at 240°C and is allowed in the area apart 3mm from the bottom of the lamp.

♦Electro-Optical Characteristics [Ta=25°C]

Symbol	Wavelength	Condition	Minimum	Typical	Maximum	Unit
V_F	735	I_F=50mA		1.85	2.30	V
	850			1.55	1.7	
I_R	660/910	V_R=5V			10	uA
P_O	735	I_F=50mA	5.0	9.0		mW
	850		5.0	9.0		
I_E	735	I_F=50mA		13.0		mW/sr
	850			13.0		
λ_P	735	I_F=50mA	710	735	755	nm
	850		835	850	865	
$\Delta\lambda$	735	I_F=50mA		40		nm
	850			40		

‡Total Radiated Power is measured by Photodyne #500
‡Radiant Intensity is measured by Tektronix J-6512.

EPITEX INC. : 4, Nishiaketa-Cho, Higashi-Kujyou, Minami-Ku, Kyoto, Japan
Tel.: ++81-75-682-2338 Fax: ++81-75-682-2267
e-mail: sales-dep@epitex.com http://www.epitex.com

A.6: Light Detector Used for NIRS

Burr-Brown Products from Texas Instruments

OPT101

SBBS002A – JANUARY 1994 – REVISED OCTOBER 2003

MONOLITHIC PHOTODIODE AND SINGLE-SUPPLY TRANSIMPEDANCE AMPLIFIER

FEATURES

- SINGLE SUPPLY: +2.7 to +36V
- PHOTODIODE SIZE: 0.090 x 0.090 inch
- INTERNAL 1MΩ FEEDBACK RESISTOR
- HIGH RESPONSIVITY: 0.45A/W (650nm)
- BANDWIDTH: 14kHz at R_F = 1MΩ
- LOW QUIESCENT CURRENT: 120μA
- AVAILABLE IN 8-PIN DIP AND 8-LEAD SURFACE-MOUNT PACKAGES

APPLICATIONS

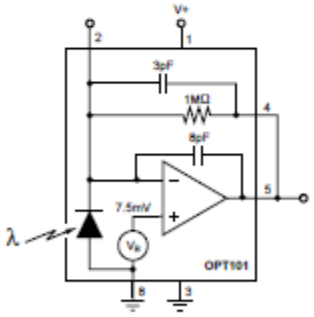
- MEDICAL INSTRUMENTATION
- LABORATORY INSTRUMENTATION
- POSITION AND PROXIMITY SENSORS
- PHOTOGRAPHIC ANALYZERS
- BARCODE SCANNERS
- SMOKE DETECTORS
- CURRENCY CHANGERS

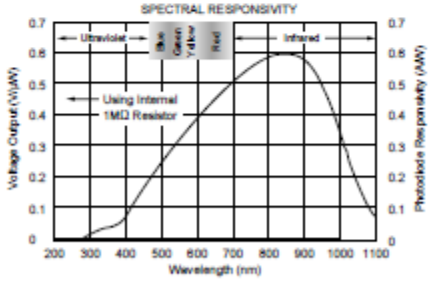
DESCRIPTION

The OPT101 is a monolithic photodiode with on-chip transimpedance amplifier. Output voltage increases linearly with light intensity. The amplifier is designed for single or dual power-supply operation, making it ideal for battery-operated equipment.

The integrated combination of photodiode and transimpedance amplifier on a single chip eliminates the problems commonly encountered in discrete designs such as leakage current errors, noise pick-up, and gain peaking due to stray capacitance. The 0.09 x 0.09 inch photodiode is operated in the photoconductive mode for excellent linearity and low dark current.

The OPT101 operates from +2.7V to +36V supplies and quiescent current is only 120μA. It is available in clear plastic 8-pin DIP, and J-formed DIP for surface mounting. Temperature range is 0°C to +70°C.





A.7: Servo Motor Used for Test Arm

Technical Specifications

Kit Contents	(1) VEX servo motor (1) VEX servo motor gear set (4 total gears, 1 replacement for each gear inside) (2) 6-32 x 1/4" Screws (2) 6-32 x 1/2" Screws (1) Clutch (1) Clutch post (1) Inventor's Guide Insert
Downloads & Docs	Inventor's Guide - Servo Inventor's Guide - Motion CAD Models of all motion components can be found on the VEX Wiki.
Compatibility	All VEX Microcontrollers. All VEX Square Shafts 0.125" (3.2mm)
Specifications	Rotation: 100 degrees Stall Torque: 6.5 in-lbs Voltage: 4.4 - 9.1 Volts (Motor life will be reduced operating outside this range) PWM Input: 1ms - 2ms will give full reverse to full forward, 1.5ms is neutral Black Wire: Ground Orange Wire: Power White Wire: PWM signal Current Draw: 20mA to 1.5 A per Servo Note: Performance varies slightly due to variations in manufacturing.
Weight	VEX continuous rotation motor 0.096 lbs (43.5 grams) 6-32 x 1/4" Screw 0.0014 lbs (0.617 grams) 6-32 x 1/2" Screw 0.00209 lbs (0.948 grams) Clutch 0.007 lbs (3.17 grams) Actual weight one one item (no packaging)