A NOVEL PRINCIPAL AND INDEPENDENT COMPONENT ANALYSIS PREPROCESSING TECHNIQUE FOR NEURAL NETWORK CLASSIFICATION OF ELECTROENCEPHALOGRAPHY SIGNALS FOR BRAIN COMPUTER INTERFACE DEVELOPMENT

by

Tyler Major

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2018

Approved by:

_____

Dr. James Conrad

_____

Dr. Yogendra Kakad

_____

Dr. Courtney Smith-Orr

_____

Dr. Maciej Noras

ABSTRACT

TYLER MAJOR. A Novel Principal and Independent Component Analysis Preprocessing Technique for Neural Network Classification of Electroencephalography Signals for Brain Computer Interface Development. (Under the direction of DR. JAMES CONRAD)

The field of brain computer interfaces (BCI) is growing rapidly. Innovations that help benefit disabled persons is the overall goal of the research, currently. Every brain computer interface consists of three basic parts: a sensing device, signal processing, and an actuator. This work contributes to the second of the three parts, signal processing. This work presents and tests a novel method for combining the already established work of principal component analysis, independent component analysis, and artificial neural networks to generate a brain computer interface for controlling a robotic hand. The sensing device is substituted by a data set and the actuator is substituted by using a simulator. This work also presents a framework for rapid development using this method and testing inside the simulated environment with different hardware to ease the transition from the theoretical to the practical.

Results of the developed algorithm were assessed with current state of the art techniques and was found to be competitive or more robust than other techniques. The algorithm was evaluated across 10 subjects, with typical results from one subject presented. Imagined left and right hand grasp intent were classified, along with another classifier for neither intent.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. James M. Conrad for his help and guidance throughout this research and my college career as well as Dr. Yogendra Kakad, Dr. Courtney Smith-Orr, and Dr. Maciej Noras for serving on my committee. I would also like to thank Dr. Stephen Kuyath and Dr. Thomas Weldon for once being parts of my committee during this process.

Foremost, I would like to thank my fiancée, Natassia, and my family whose support and encouragement has allowed me to make it this far.

TABLE OF CONTENTS

LIST OF FIGURES viii

LIST OF TABLES xii

LIST OF ABBREVIATIONS 1

CHAPTER 1: INTRODUCTION 1

1.1. Motivation 1

1.2. Objective of this Work 1

1.3. Contribution 2

1.4. Organization 3

CHAPTER 2: THEORY AND BACKGROUND 4

2.1. Introduction 4

2.2. Practical Applications 4

2.3. Signal Acquisition 6

2.3.1. Introduction 6

2.3.2. Functional Magnetic Resonance Imaging (fMRI) 6

2.3.3. Near-Infrared Spectroscopy (NIRS) 8

2.3.4. Magnetoencephalography (MEG) 8

2.3.5. Electrocorticography (ECoG) 9

2.3.6. Electroencephalography 10

2.4. Filtering 12

2.4.1. Introduction 12

2.4.2. Spectra Bands 12

2.4.3. Power Spectral Density 13

2.4.4. Principal Component Analysis 14

2.4.5. Independent Component Analysis 14

2.5. Machine Learning for Classification 15

2.5.1. Artificial Neural Networks 15

2.5.2. Support Vector Machine 17

CHAPTER 3: SYSTEM FRAMEWORK 19

3.1. Overview of the System 19

3.2. Block Diagram of the System 19

3.3. Input Signal 20

3.4. Signal Processing 22

3.5. Robot Operating System 24

CHAPTER 4: MATLAB TOOLS FOR BCI PROCESSING 26

4.1. Introduction 26

4.2. Example of BCI Development in MATLAB 26

4.2.1. Event Duration 29

4.2.2. Data Signals Associated with Motor Movement 29

4.2.3. BCILAB 32

4.2.4. Online Analysis 45

CHAPTER 5: HARDWARE SYSTEMS 46

5.1. HAPTIX 46

5.2. InMoov 49

5.3. Gazebo 52

CHAPTER 6: RESULTS 55

6.1. Similar Study Results 55

6.2. Analyzing All 64 Electrodes 56

6.3. Analyzing 16 Electrodes 59

6.4. Baseline Neural Network Performance 61

6.5. Analysis of Research Results 61

6.6. Generalized Neural Network 63

CHAPTER 7: CONCLUSION 65

7.1. Future Work 66

REFERENCES 68

LIST OF FIGURES

FIGURE 1.1: Basic structure of a BCI with highlighted section showing area of contribution. 2

FIGURE 2.1: Example of a P300 Type Spelling BCI using a 6X6 Matrix of Letters and Numbers, and a Backspace. [1] 5

FIGURE 2.2: Electrical current activity in the brain generates a magnetic field. 9

FIGURE 2.3: Intracranial electrode mesh for electrocorticography electrophysiological monitoring [2]. 10

FIGURE 2.4: The 10/20 International Positions and Associated Labels [3]. 11

FIGURE 2.5: Human motor cortex containing the primary sensorimotor cortex for ERS/ERD detection of motor imagery [1]. 13

FIGURE 2.6: Neural network example showing how an input of n number of signals travels through hidden layers of neurons to choose the output with the highest probability of matching. 16

FIGURE 2.7: Examples of linear separations created by a SVM. H_1 does not separate the classes. H_2 and H_3 both separate the classes, but H_3 does so with a maximum margin [4]. 17

FIGURE 2.8: The solid line represents the maximum-margin hyperplane that maximizes the linear separation of the two trained classes. The parallel dashed lines are the hyperplanes that separate the two classes, with the intersecting test classes being the support vectors of the system [5]. 18

FIGURE 3.1: BCI system detailing signal acquisition, α and β filtering, PCA and ICA processing, an ANN classifier, and grasp detection. 20

FIGURE 3.2: System that consists of ROS acting as the middleware between the controlling algorithm on the computer running MATLAB and either the simulation environment or an actuating hand. The ROS node communicates, bidirectionally, as a handler for all signal flow. 21

FIGURE 3.3: The International 10-10 Labeling System for EEG Recording, Excluding Electrodes Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10 [6]. 23

FIGURE 4.1: Menu to change the channels read by EEGLAB. 27

FIGURE 4.2: Raw unprocessed waveform of EEG recordings from the trial. 28

FIGURE 4.3: Filtered signal that removes noise on the signal. 28

FIGURE 4.4: Green area highlighting duration of a typical target that is presented to a subject. This target was present from 24.5 - 28.5 seconds of the trial. 30

FIGURE 4.5: Picture showing the relevant sections of the brain for motor movements. 32

FIGURE 4.6: Weighted ICA of one participant's trials of hand movements. This is effectively a heat graph, with higher intensity areas colored in red, and lower intensity areas transitioning to blue. This was generated with the help of the EEGLAB plugin for MATLAB. 33

FIGURE 4.7: Main menu of the BCILAB plugin for MATLAB. 34

FIGURE 4.8: Menu and options for loading the dataset file into BCILAB. 34

FIGURE 4.9: Window for choosing the approach for the model. Common spatial pattern approach shown. 35

FIGURE 4.10: Option for CSP approach. 36

FIGURE 4.11: Calibration options for generating a model. 38

FIGURE 4.12: Results from the default parameters of the CSP model approach. 39

FIGURE 4.13: Results from using the generated CSP model on the imagined movement data. 40

FIGURE 4.14: Approach of spectrally weighted CSP designed specifically for imagined movements. 41

FIGURE 4.15: Configurations for the spectrally weighted CSP approach. 42

FIGURE 4.16: Results of the generated spectrally weighted CSP on the calibration data. 43

FIGURE 4.17: Final performance results of the spectrally weighted CSP model on imagined movement. 44

FIGURE 4.18: Visualization of spectrally weighted CSP model generated and the contributions of each frequency for each pattern that emerged. 45

FIGURE 5.1: Example simulation environment of the DARPA HAPTIX robotic arm. This simulation includes a graphical feedback of the touch sensors on each hand as well as a list of the objects in the scene that can be manipulated [7]. 47

FIGURE 5.2: Schematic describing the location of the motors in the hand. The numbers on the diagram correspond to the index position of the array in the C API of each motor. Each motor provides feedback on its position, torque, and velocity [7]. 48

FIGURE 5.3: Location of contact sensors of the DARPA HAPTIX hand. The numbers on the diagram correspond to the index position of the array containing the sensors. There is also another array that contains the IMU data (located under the fingernails) to be used as additional sensor feedback [7]. 49

FIGURE 5.4: 3D printed had of the InMoov project that showcases a closeup of the mechanics of finger and joint location. Each servo, connected to each joint, provides 90 degrees of movement, enabling the hand to fully close. 50

FIGURE 5.5: Forearm of the arm unit of the InMoov. While normally covered, this shows the tensor strings that connect up to the hand and the servos that pull to close the fingers. 51

FIGURE 5.6: Highlight of the wrist movement capabilities of the InMoov arm. The wrist is capable of rotating 180 degrees, independent of the finger movement. 52

FIGURE 5.7: Turtlebot2 placed in a simulated environment, powered by Gazebo. Gazebo comes preloaded with example physics objects of a wide variety that are already detailed and properly scaled for the simulator. 53

FIGURE 5.8: IP message communication flow between MATLAB running on a physical machine and Gazebo running on a virtual machine. ROS handles the communication by connecting these applications through a Uniform Resource Identifier (URI) to label the applications with their IP addresses. As is shown, the ROS_MASTER_URI runs on both machines and handles the data flow. 54

FIGURE 6.1: 16 Electrodes over the motor cortex used to detect movement intent for study. 59

FIGURE 7.1: System flow of final design for deployment. The training algorithm is fed in to MATLAB with the online, real-time, signals as the input. The algorithm classifies the signal as either a left or right hand grasp and sends a signal to the ROS core [8]. This is then either sent to the simulator [9] or a prosthetic limb [10]. 67

LIST OF TABLES

TABLE 6.1: Comparison of Similar Methods 55

TABLE 6.2: Sample of Results with Processing 64 Starting Electrodes 58

TABLE 6.3: Sample of Results with Processing 16 Starting Electrodes 60

TABLE 6.4: Comparing Neural Network Performance with no Technique 61

TABLE 6.5: Comparing Results with No Dimensionality Reduction on Different Number of Starting Electrodes 62

TABLE 6.6: Comparing Dimensionality Reduction with Similar Starting Dimensions 63

TABLE 6.7: Generalizing the BCI Neural Network 64

CHAPTER 1: INTRODUCTION

Brain computer interfaces (BCI) are a relatively new technology that takes advantage of the innate computing power of the brain. Developing BCI have, up until recently, been thought of as science fiction. Ever since the first discovery of electroencephalography (EEG) by Berger, scientists have been trying to decode signals from the brain [11]. Generally, it can be said that a BCI is a system where there is a direct communication pathway between the brain and an external device. For such a system to be viable, it is imperative that interpreting signals from the brain is handled in a safe and timely manner. This consists of many clinical trials with sensing hardware and exhaustive testing of software interactions.

1.1 Motivation

Brain computer interfaces involve collaborative communication from a human brain and the actuating system. Normally the actuating system is the human body i.e. legs, arms, hands, and fingers. When that pathway of communication is damaged or lost in some way then control of those systems is no longer possible. In order to regain function, these neurological connections must either be repaired or augmented. One such, popular, area of research focuses on patients with amyotrophic lateral sclerosis (ALS), also known as "locked-in" syndrome [12–18].

1.2 Objective of this Work

In this paper, the primary focus is to design and develop a simulation environment using the Robot Operating System (ROS) to enable the testing, simulation, and performance evaluation of different classification techniques for BCI development. BCI also have to be trained on each individual separately; that is to say that one

trained classifier can not be used from one person to the next without heavy retraining. This scheme could also hopefully cut down on the time one would have to spend training by abstracting the evaluation and also brining in a generalized model of one person's trained data and trying to apply that to another individual.

1.3 Contribution

The main contribution of this work is to provide an improved model for classification of mental tasks and validation through a simulated environment. Special consideration will be given to reduce the false positive classification from prior work. The combination of these directives will move the field of work closer to consumer use of brain computer interfaces. Figure 1.1 highlights the area of contribution for this work in the overall scheme of BCI development. Rather than focusing on sensing signals, which would require metamaterial knowledge, or end user applications, which would require clinical trials and mechanical focus, this work contributes to bridging the gap between the two ends. Previous work in this area is expanded upon and new combinations of techniques are presented that contribute to the overall field.

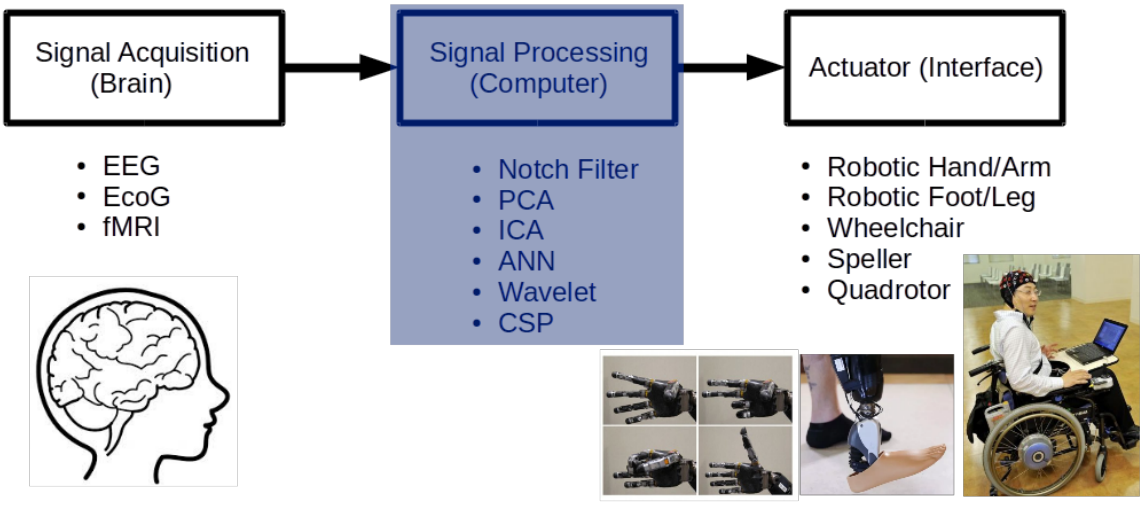


Figure 1.1: Basic structure of a BCI with highlighted section showing area of contribution.

1.4 Organization

This dissertation is divided into six chapters. Chapter 2 reviews the theory and background of the work such as, practical applications, signal acquisition techniques, various filtering methods, and different machine learning algorithms. Chapter 3 describes the system framework and each individual step with how it handles and modifies the signals it is given. Chapter 4 summarizes the MATLAB tools available to researchers for developing a BCI and covers an example of a development process [19]. Chapter 5 covers the hardware systems that this BCI can control for grasping applications. Chapter 6 discusses the results of the research. Chapter 7 concludes the document with final remarks and plans the pathway for future work of the research.

CHAPTER 2: THEORY AND BACKGROUND

In this chapter we will review the basic concepts of

- Brain Computer Interfaces (BCI) with examples
- Signal Acquisition techniques and hardware
- Filtering principles for separating signals
- Machine Learning that is starting to be used in the field

2.1 Introduction

Generally, it can be said that a BCI is any system that incorporates a direct communication pathway between a brain and an external device. A BCI traditionally consists of four main parts; a sensing device, an amplifier, a filter, and a control system.

2.2 Practical Applications

The first proposed application of a BCI was for use in therapeutics and for mental disorder classifications [20, 21]. Modern BCI research focuses on patients with amyotrophic lateral sclerosis (ALS), also known as "locked-in" syndrome [12–18]. BCI research has also expanded to include systems that healthy individuals can utilize to expand normal human capabilities [22, 23].

BCIs are categorized into two different types: dependent and independent. A dependent BCI relies on element pathway in the brain to generate activity. An example of this BCI type would be the spelling program shown in Figure 2.1 [24].

This system is monitoring the brain waves for event related potentials (ERP), recognizable patterns in brain waves that occur during stimuli, such as being presented

Figure 2.1: Example of a P300 Type Spelling BCI using a 6X6 Matrix of Letters and Numbers, and a Backspace. [1]

with a specific image or an imagined movement. A matrix containing the desired outputs, such as an array of letters in a spelling program, flashes at a specific rate. Utilizing the data from the flashing and the timing of the ERP, the desired letter is extrapolated. The specific ERP that is being monitored in the case of the spelling program is called a visual evoked potential (VEP). The contribution from the visual cortex is the dominant signal in VEPs, so naturally this signal is used to determine which letter the subject is observing. The dependent BCIs are accurate and commonly used, however this model is inadequate for a person with severe neuromuscular disabilities since the signal is derived from an extraocular muscles in this case.

An independent BCI does not depend on any of the normal pathways in the brain for the output. One example is the utilization of the flashing matrix of letters idea discussed previously, but looking for a different identifying signal. This study is looking for a signal, called a P300 evoked potential, produced by the person that corresponds to a specific flashing letter [25, 26]. For this example the output EEG signal generation is based on intent and not eye orientation. This is the preferred area of theoretical research as the brain makes new pathways to control an output. This is a great advantage to patients with disabilities who lack normal output pathways, such as a patient with ALS.

2.3 Signal Acquisition

Signal acquisition is a substantial challenge in the field of BCIs. Each method of capturing signals from a brain has its own strengths and weaknesses for recording different portions of the signals. End use, intended by the designer, is the factor that filters out which method to move forward with.

2.3.1 Introduction

In medical applications where noise is a concern, such as measuring seizure patients for temporal biopsy, an electrode mesh is placed directly on the brain's surface; this method is known as electrocorticography (ECoG). This has also been adapted to the field of BCIs [27] by taking advantage of the two-week window that the mesh is implanted, identifying the portion of the brain to removed to reduce or eliminate seizures. More widely available, and much safer due to being noninvasive, is the use of an EEG recording cap to capture the electrical impulses of the brain. The drawbacks of this method are attenuation of the signal and deflection. To give a better idea of how this effects the signal a typical amplitude of a scalp measurement is about 5-200 μV while the same measurement will be about 1-2 mV when measured on the surface of the brain. This is the reason why there needs to be processing performed on the signal to filter out the ambient noise inherent in the signal.

2.3.2 Functional Magnetic Resonance Imaging (fMRI)

One of the more practiced methods for detecting neurological activity for research purposes is called Functional Magnetic Resonance Imaging (fMRI). This process involves observing a subject's change in blood flow (hemodynamic response) while they are laying in a Magnetic Resonance Imaging (MRI) machine. The response that active neurological processes produce is known as the Blood Oxygen Level Dependency (BOLD) [28]. This response arises from the basic principal that regions of the brain that are more neurologically active will require a higher hemodynamic response than

areas of the brain that are not engaged. One of the main drawbacks of using fMRI is the relatively slow reaction time of the system. This delay is attributed to the response time of the BOLD response of the brain which typically can delay anywhere from 3 to 6 seconds [29]. However, there is research that suggests that this delay can be overcome with techniques that look for finer BOLD responses in specific areas and using that information for a real-time BCI or as an initial guide for fine tuning EEG procedures [30].

The drawbacks of an fMRI do not exclude it as usable and viable technique for BCI control. The BOLD signal response has successfully been used as an indicator for intended movement [29]. A study placed participants in an MRI machine that showed high variance between BOLD responses for different intended movement. This analysis was performed off-line, but clearly shows the feasibility of using fMRI for an on-line BCI. This study occurred with four volunteers and consisted of two calibration sessions and a feedback session. During the feedback section the volunteers were shown the activity levels through a video projection of the regions of interest (ROIs). A custom developed software that ran separately on another computer made this process possible. During the experiment it was very important, as it is with all BCIs, to remove artifacts, such as background noise from eye and muscle movement a.k.a. electromyography (EMG), would override the desired signals. Real-time motion correction was used to remove the contributions from muscular movement.

Another example of fMRI use for brain control method is the detection of imagined and executed unimanual and bimanual movements [31]. In this experiment eight healthy right-handed volunteers were chosen to participate. Their handedness was verified using the Edinburgh Handedness Inventory [32]. The experiment consisted of three parts: two unimanual movement and one bimanual movement. Subjects were asked to individually move their fingers, excluding thumbs, in predefined repeating sequences. Once the sequence was completed, the trial was performed again with

imagined movements as opposed to actual movements. While there was predicted variability in each individual subject, there was also a clear trend. Actual movements were consistently at a higher potential level than imagined movements, and thereby providing a more accurate signal, but the imagined movements still provided a cluster of neurons acceptable for reliable signal detection.

2.3.3 Near-Infrared Spectroscopy (NIRS)

NIRS is a method that uses light close to the infrared spectrum to monitor a response that is similar to the BOLD response called regional cerebral blood flow (rCBF) [33]. NIRS is used to look over a general area of the brain for activity, though LEDs have been used for more precise detection. Pairs of illuminators and detectors form channels for the signals. Near-infrared rays emit from each illuminator and pass through the skull and brain tissue to be received by the detectors. An example of NIRS being used for an on-line BCI spelling program can be found [33]. This study involved five individuals who underwent a baseline trial, a partition, and a motor task. The motor task involved finger tapping which would be dictated by on screen prompts. One of the weaknesses of NIRS measuring is the dependency of passing through the skull; this means that things like hair can greatly hamper the signals and give faulty readings.

2.3.4 Magnetoencephalography (MEG)

MEG provides more sensors, and thus more spatial information, than traditional a EEG. In order to take MEG recordings, a subject, in a magnetically shielded room, is placed in a chair with an array of superconducting quantum interference devices (SQUIDs) around their head as the magnometer. The obvious drawback to this approach is the dependency of a magnetically shielded room and a large machine to sense the brainwaves. Research has proven, even with these constraints that MEG is still a viable and reliable enough of a method to be explored further [34].

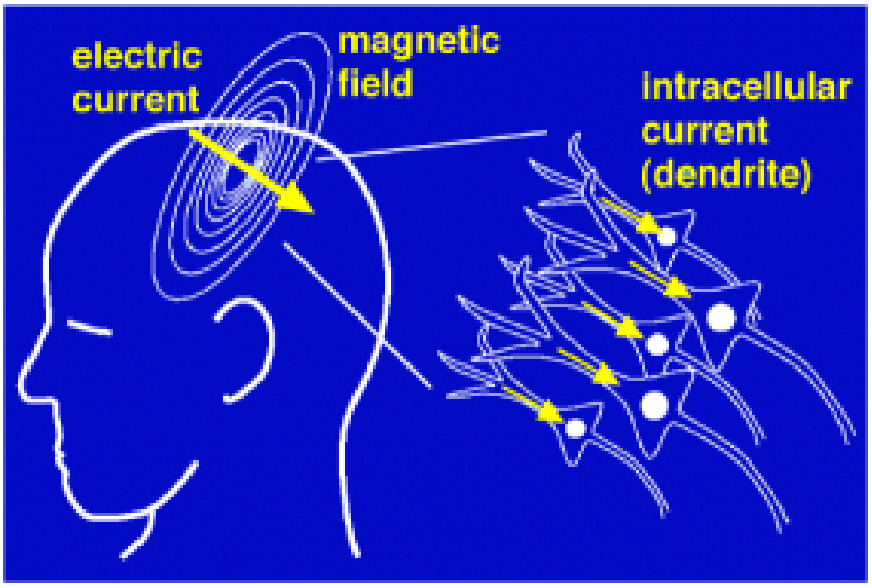


Figure 2.2: Electrical current activity in the brain generates a magnetic field.

2.3.5 Electrocorticography (ECoG)

Differing from the previous methods ECoG is an invasive method. ECoG requires surgery to implant electrode pads directly onto the surface of the brain to receive signals from the cerebral cortex. The advantages of this are immediately clear: high spatial resolution, broad bandwidth, high amplitude, and less vulnerability to EMG [27]. ECoG is also widely used as an identifier for the localization of epilepsy focal points. An array of 64 electrodes is implanted onto a portion of the brain called the epileptic focus to identify the part of the brain that should be removed by resection surgery.

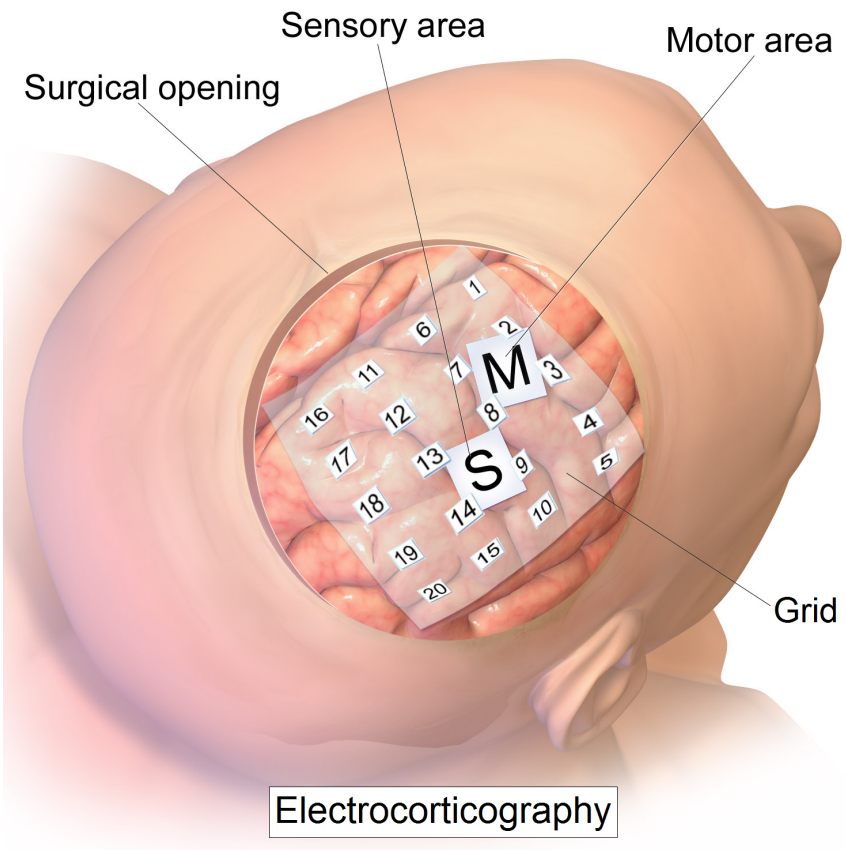


Figure 2.3: Intracranial electrode mesh for electrocorticography electrophysiological monitoring [2].

During one study, patients with epilepsy were implanted with these electrodes [27]. In the period of the one to two weeks that the electrodes are recording data to localize the seizure area, researches used the electrodes to generate a BCI. While the electrodes were removed in this instance due to the epileptic nature of the patients, the success of this study proves that these arrays are a valid method for use in BCI development and not just epileptic identification.

2.3.6 Electroencephalography

EEG is a technique involving the placement of electric field sensing electrodes around the scalp of an individual. The placement of these electrodes is standardized with a technique called 10/20 positioning [6]. A subject is instructed to clean their hair vigorously the night before the readings are taken. Measurements are taken

according to the international 10/20 manual and the electrodes are placed against the scalp of the subject with a conductive medium, such as conductive gel, placed on the pads to facilitate the acquisition of signals.

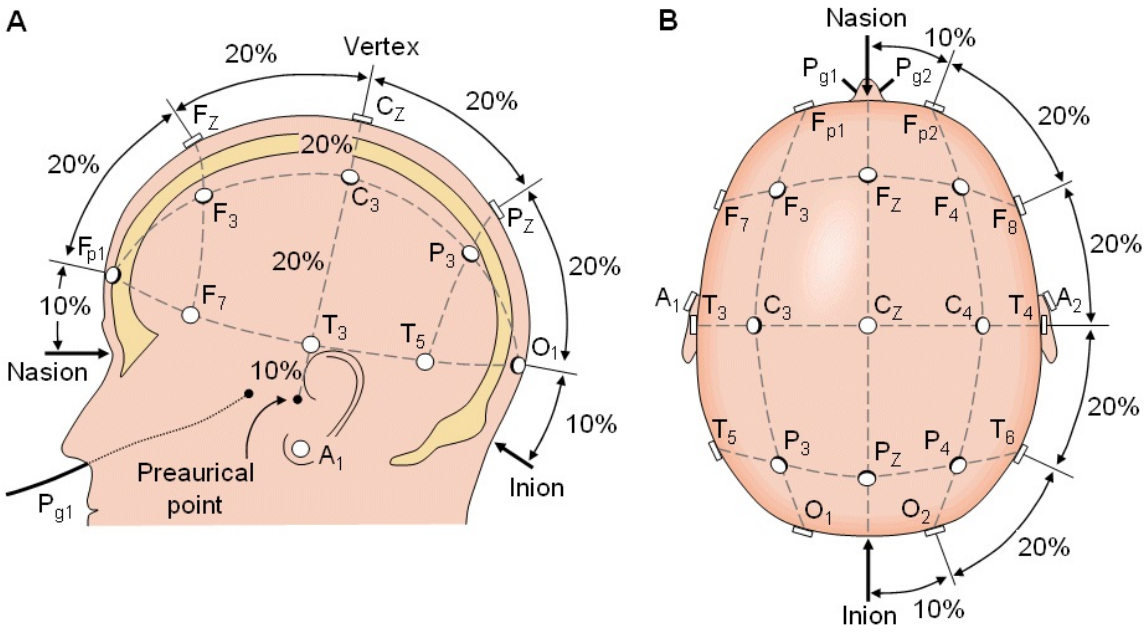


Figure 2.4: The 10/20 International Positions and Associated Labels [3].

This method is, by far, the most popular for capturing signals from the brain. A few of the factors that make EEG such an attractive method are as follows: standardization of electrode placement, information on acquisition techniques well documented and plentiful, established as a reliable method with known filtering techniques, and the relatively low cost compared to other methods.

Since this is the most popular method, there is ongoing research to simplify the use of EEG for commercial applications, rather than the often complex and time consuming task of applying the 10/20 system [35]. Systems such as these are predicted to be the on the user end of a BCI as opposed to the research end. These types of headsets are attractive to the user for their ease of set up and very low calibration times; with this example being in the range of ten seconds. Another big advantage to this system is that it can potentially work with many different types of headsets

that are already available in the market.

One big caveat about EEG signals through, is that a sensed signal does not necessarily mean that that electrode is the source of the signal [36]. This counter intuitive phenomena is due to the fact that the brain is folded. In order to find the source of the signals, methods such as independent component analysis (ICA) are being widely used now as a localization technique. EEG signals are thus classified as being dipolar, meaning that in representations it is shown that there is a signal and an associated direction for the propagation of that signal [37].

2.4 Filtering

2.4.1 Introduction

To solve the problem of a noisy signal, filtering must be used to reduce that noise and extract the underlying signal features. Such filtering requires an understanding of EEG signals and the challenges that must be overcome by using this sensing technique, and a grasp of the expected signals.

2.4.2 Spectra Bands

A common way to differentiate between the activity in the brain is to separate it into sensorimotor rhythm (SMR) bands, shown in Figure 2.5. In these frequency ranges the α and β bands, 8-12 Hz and 13-25 Hz, respectively, are closely related to sensorimotor processes. The onset of a voluntary movement causes a desynchronization between the α and β bands, refereed to as an Event Related Desynchronization (ERD). Once the movement concludes then there is an increase in the overall power between the two bands, refereed to as an Event Related Synchronization (ERS). Using this concept it is possible to identify intended movement in patients that have lost motor functionality. The motor imagery can be identified by measuring the α and β bands with sensors over the primary sensorimotor cortex.

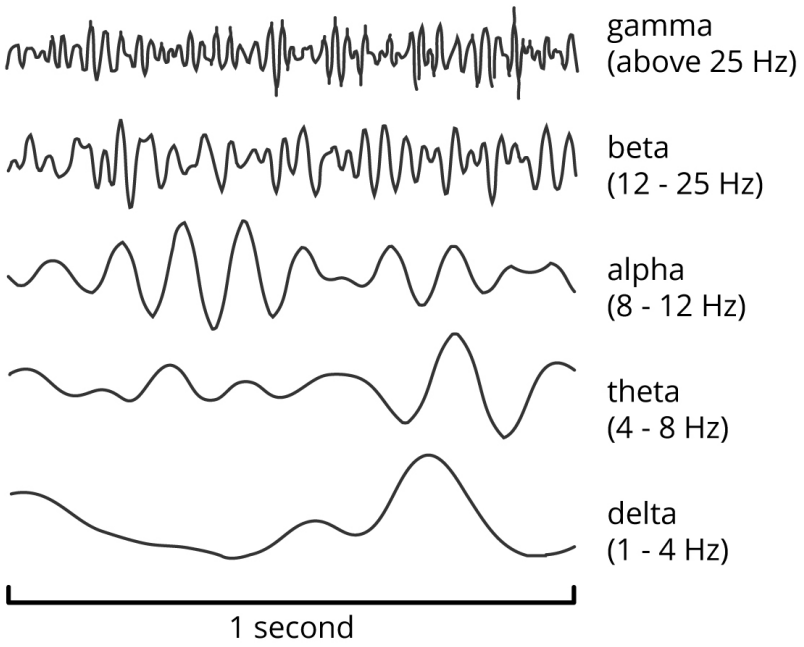


Figure 2.5: Human motor cortex containing the primary sensorimotor cortex for ERS/ERD detection of motor imagery [1].

2.4.3 Power Spectral Density

Power Spectral Density (PSD) describes the distribution of power in a physical signal when decomposed over a frequency spectrum. This is useful in finding the frequency characteristics of a windowed event in a finite time interval. Using the PSD on a signal sampled in discrete time yields an estimate based on summation, rather than integration. In order to generate a more accurate estimate of the true PSD of the function it is necessary to average these estimates over many trials. This is commonly refereed to as a periodogram, which converges to the true PSD as the number of estimates approaches infinity. It is important to first reduce the dimensionality of the signal through Principal Component Analysis (PCA) and Independent Component Analysis (ICA) to greatly reduce the computations needed in calculating the PSD. The PSD only need be ran on independent components.

2.4.4 Principal Component Analysis

Principal Component Analysis is a statistical transform used in data analysis to decompose correlated variables into uncorrelated variables, called principal components. In BCI applications this has the use of reducing dimensionality and redundancy in data.

PCA learns a linear transform $h = f(x) = W^T x + b$ with input $x \in IR^{d_x}$, where the columns of $d_x \times d_h$ matrix W form an orthogonal basis for the d_h orthogonal directions of greatest variance in the training data [38].

By taking a dataset and separating the components into differing principal components, the leading eigenvectors of the covariance matrix, it is possible to use only the principal components that have the greatest variation and prune the least represented eigenvectors. PCA has the assigns the largest variance as the principle component. In this sense, PCA is a tool used to help reduce dimensionality and retain the feature information, thereby improving classification. This method is also closely tied to Linear Discriminant Analysis (LDA) since they both seek linear combinations of variables which best explain the data.

2.4.5 Independent Component Analysis

In order to use ICA there are three criteria that the signals must follow:

1. The number of ICs are less than or equal to the number of observed signals
2. The artificial and cerebral sources are linearly mixed and statistically independent
3. Propagation delays through the missing medium are negligible

Since EEG signals fall under the purview of these conditions [39] it is possible to

try and solve for $x_1, ..., x_n$ of n linearly mixed components by

$$x_j = a_{j1}s_1 + a_{j2}s_2 + ... + a_{jn}s_n, forall j. \tag{2.1}$$

The random variables $\mathbf{x}$ and $\mathbf{s}$ are said to be latent variables in the mixed signal. These signals are generally represented in vector-matrix notation where $\mathbf{x}$ and $\mathbf{s}$ are vectors and the elements a_{ij} are represented by the matrix $\mathbf{A}$. In ICA the vectors are column vectors. The general form of the mixing model is

$$x = As. \tag{2.2}$$

2.5 Machine Learning for Classification

This section discusses the leading machine learning algorithms used in the field. Once EEG signals are sufficiently filtered then it is necessary to use those signals to extract the underlying patterns in the data that suggests intent. These patterns are typically hard to describe programmatically, which necessitates the use of other tools to classify this information.

2.5.1 Artificial Neural Networks

Artificial neural networks (ANNs) is a computational model used in machine learning that builds a complex connection of simple artificial neurons to mimic the structure of a biological brain. Without the use of back propagation this technique is strictly a unidirectional process; input signals propagating through hidden layers to arrive at outputs. The input signals travel through these hidden layers based on how statistically likely they match the neuron; if the match is close enough then the neuron activates and the signal propagates by the neuron activating. This is referred to as the threshold or limiting function and is typically a value between 0 and 1. Figure 2.6 demonstrates how a set of inputs is cascaded through a set of n hidden

layers to determine an output.

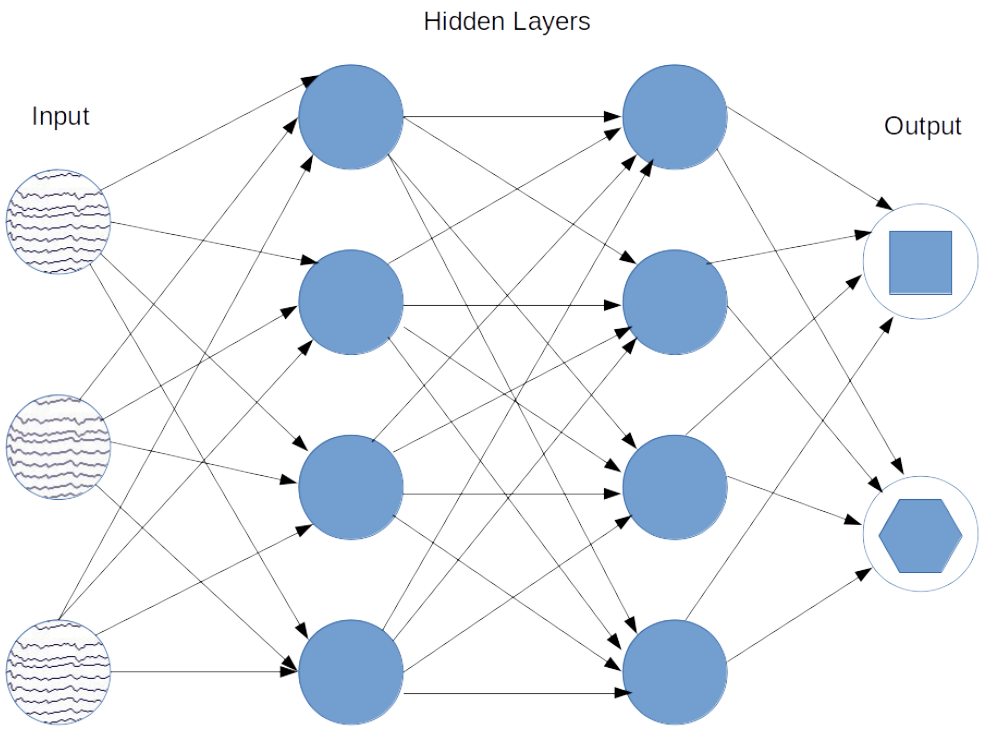


Figure 2.6: Neural network example showing how an input of n number of signals travels through hidden layers of neurons to choose the output with the highest probability of matching.

This function is generally trained in BCI applications through back propagation; comparing a set of inputs and modifying the thresholds, using the gradient of a loss function, based on a known outcome. In this way the threshold is trained and, hopefully, optimized by the system. An important aspect of back propagation is that the intermediate hidden layers self-organize themselves in such a way as to try and characterize the entire input space. This is to say, the hidden layers will order such that a new input, even if incomplete or containing noise, will be recognized if it contains features that resemble the trained data.

2.5.2 Support Vector Machine

Support Vector Machines (SVM) is a supervised learning model used for classification and regression analysis. SVM is a non-probabilistic binary linear classifier, meaning the input of training examples is labeled as belonging to one of two categories to build a model of for new examples. In cases where data are not labeled, an improved method to SVM is used, called Support Vector Clustering (SVC). SVC can be used either when data is not labeled or is labeled incompletely.

Generally, a SVM is modeled as a hyperplane, or set of hyperplanes, that seeks to maximize the separation in labeled data points, referred to as classes. Figure 2.7 illustrates how this margin is modeled.

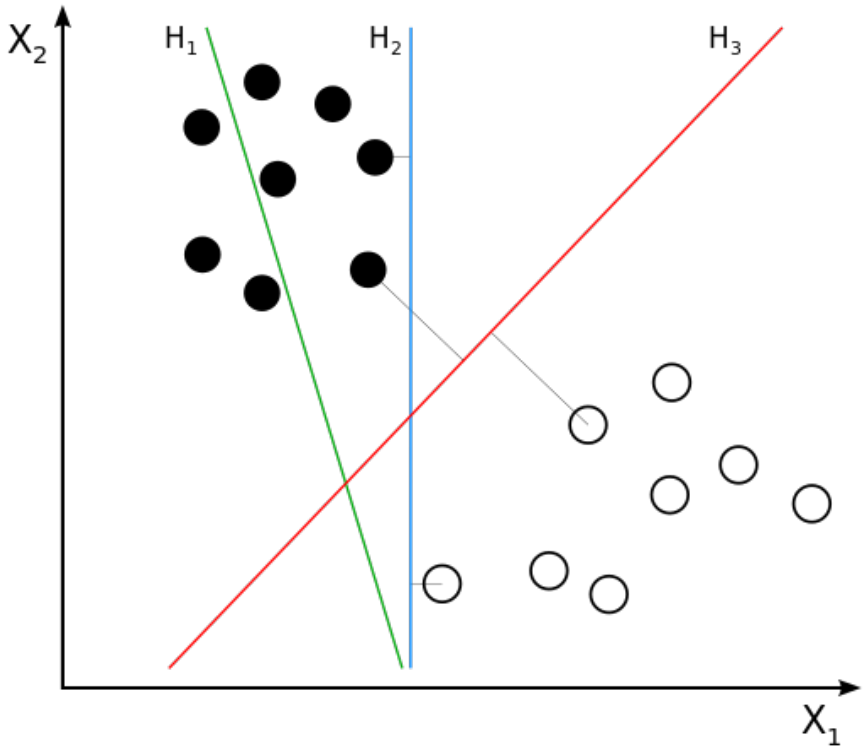


Figure 2.7: Examples of linear separations created by a SVM. H_1 does not separate the classes. H_2 and H_3 both separate the classes, but H_3 does so with a maximum margin [4].

Assuming the training data is linearly separable, two parallel hyperplanes can be selected that have the maximum distance between them. Strictly, this line is a

maximum-margin hyperplane and the samples on the parallel hyperplanes constitute the support vectors.

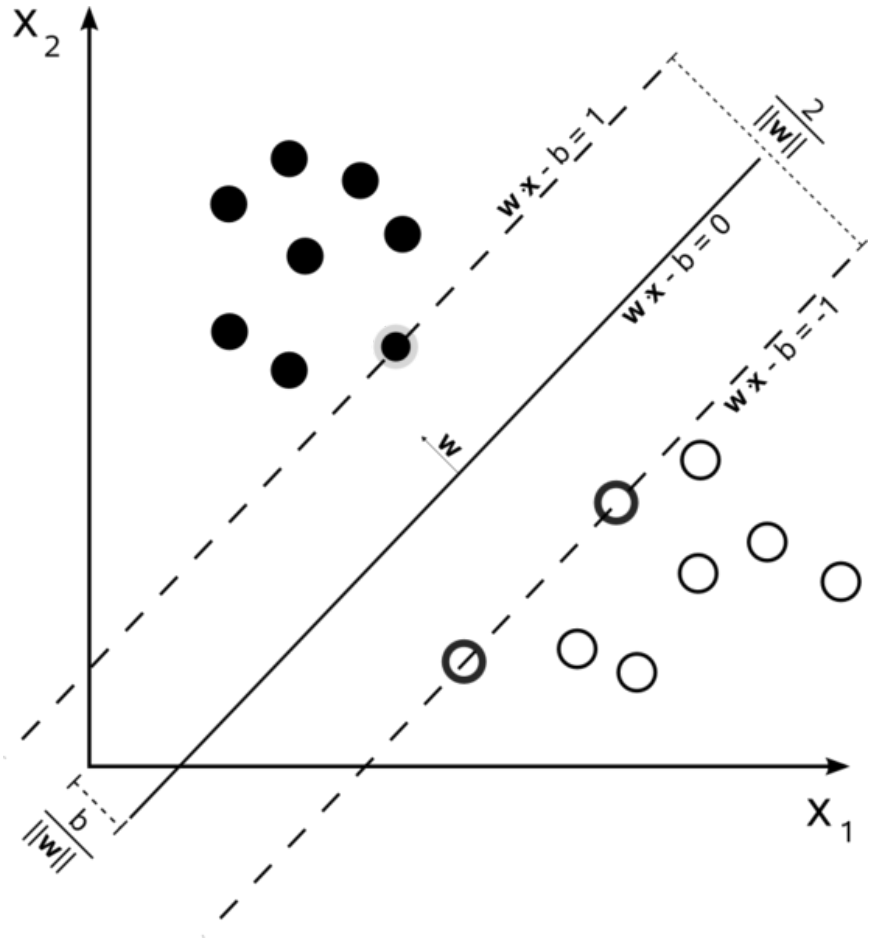


Figure 2.8: The solid line represents the maximum-margin hyperplane that maximizes the linear separation of the two trained classes. The parallel dashed lines are the hyperplanes that separate the two classes, with the intersecting test classes being the support vectors of the system [5].

CHAPTER 3: SYSTEM FRAMEWORK

3.1 Overview of the System

This method differs from current methods in the field through use of PCA processing before ICA processing and the use of a back-propagation neural network classifier. While ICA is used extensively in the field, it has, at the time of this research, never been paired with PCA.

Figure 3.1 shows the architecture of the proposed method. The signals are acquired through the use of an EEG cap fitted with electrodes or a dataset. The signals are pre-filtered and are input in to the PCA and ICA processing. Once that stage has completed and the signals are framed properly then the neural network will train and classify the signals as either a left or right hand grasp, or neither if the correct signal is not present. The remainder of this chapter covers each stage of Figure 3.1 in greater detail.

3.2 Block Diagram of the System

The price and availability of neurally linked prosthetics poses challenges to researchers in validating control and feedback schemes for practical, everyday, use. Until a standard prosthetic emerges then it is to the benefit of the community to try and use frugal means to make their solution more widely applicable. This has the added benefit of a wide variety of systems being developed to exhaustively determine the appropriate hardware and software reticulation. The proposed system diagram, simplified, can be seen in Figure 3.2. This consists of the Robotic Operating System (ROS) as a communication middleware between the BCI developed here and an end product; namely a prosthetic hand or some simulation software.

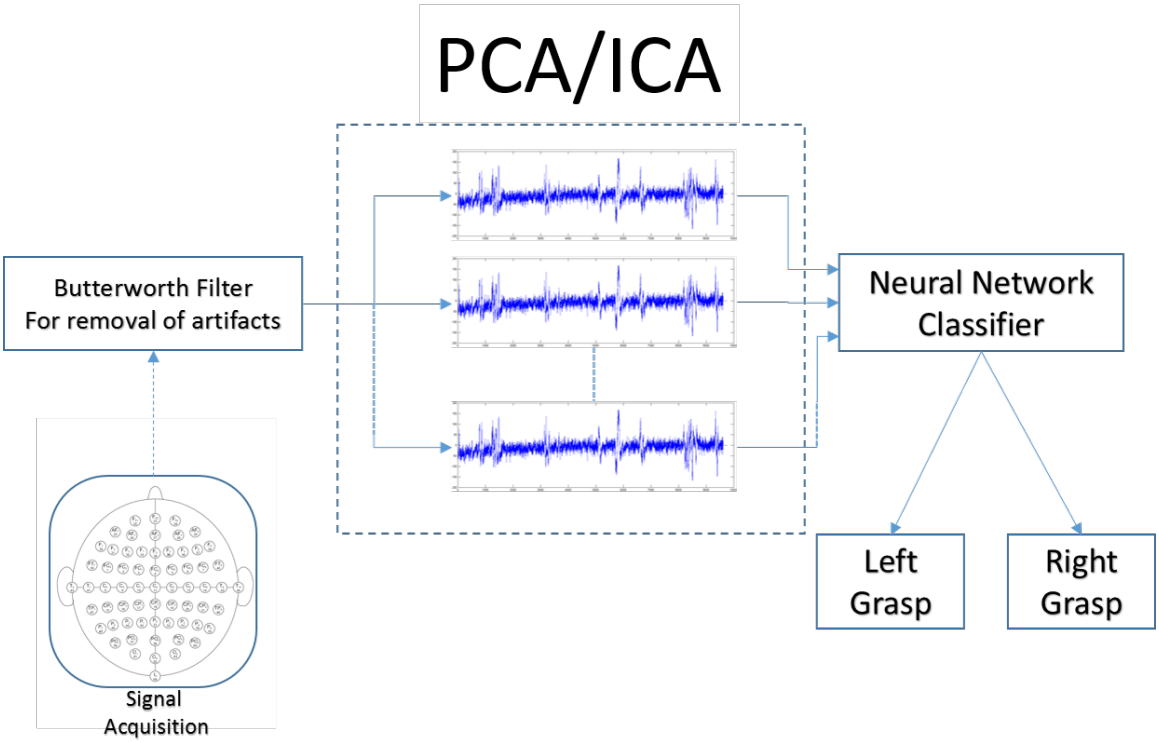


Figure 3.1: BCI system detailing signal acquisition, α and β filtering, PCA and ICA processing, an ANN classifier, and grasp detection.

3.3 Input Signal

The dataset used in this study was obtained from the open source readings from [40] that were captured using the PhysioNet toolkit though the BCI2000 software [41]. Understanding how these readings were taken and what the signals mean is paramount to understanding the BCI that was generated in this study. The total dataset contains over 1500 one minute and two minute EEG recordings from 109 participants. The subjects were instructed to perform different motor and imagery tasks while the EEG data were recorded using the BCI2000 system. Every subject performed the same 14 tasks. These tasks included two one minute runs, one with eyes open and one with eyes closed, that were used to perform baseline readings of the participants resting EEG functions. They then performed three two minute runs of four tasks. These experiments included performing the following tasks three times each:

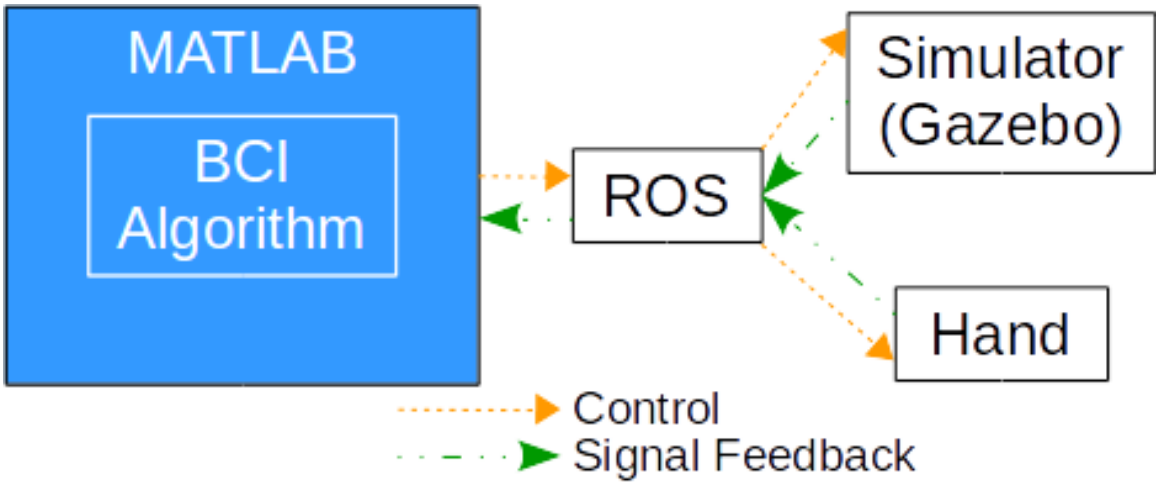


Figure 3.2: System that consists of ROS acting as the middleware between the controlling algorithm on the computer running MATLAB and either the simulation environment or an actuating hand. The ROS node communicates, bidirectionally, as a handler for all signal flow.

1. A target appeared on either the left side or the right side of the screen. The subject was instructed to open and close the corresponding fist (left fist for the target on the left side of the screen and the right fist for the target on the right side of the screen). After the target disappeared the subject relaxes.

2. A target appeared on either the left side or the right side of the screen. The subject was instructed to only imagine opening and closing the corresponding fist (left fist for the target on the left side of the screen and the right fist for the target on the right side of the screen) and to not physically open and close their hand. After the target disappeared the subject relaxes.

3. A target appears on either the top or the bottom of the screen. The subject was instructed to open and close both fists if the target appears on the top of the screen or to flex and relax both feet if the target appears on the bottom fo the screen. After the target disappeared the subject relaxes.

4. A target appears on either the top or the bottom of the screen. The subject was instructed to only imagine opening and closing both fists if the target appears

on the top of the screen or to only imagine flexing and relaxing both feet if the target appears on the bottom fo the screen. After the target disappeared the subject relaxes.

The order in which these experiments were performed were: baseline with eyes open, baseline with eyes shut, task 1, task 2, task 3, task 4, task 1, task 2, task 3, task 4, task 1, task 2, task 3, and task 4 for a total of 14 experiments. The open source data from these trials are provided in EDF+ format which each contain 64 EEG signals sampled at 160 samples per second. The file also includes and annotation channel to show when the targets were presented on the screen for the individuals for each trial. The study also includes .event files that contain identical data for use with the PhysioToolkit software. The annotations for the events consist of three separate states: rest, target for either the left fist or both fist movement for the appropriate runs, and target for the right fist or both feet for the appropriate runs.

In order to capture the EEG signals the team used a 64 electrode cap in the international 10-10 layout (excluding electrodes Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10) as shown in Figure 3.3.

It is important to note that the signals recorded with the software are numbered from 0 to 63, though the numbers of Figure 3.3 are labeled from 1 to 64. For example the electrode number 33 in the software recording is in reference to the electrode labeled 34 in Figure 3.3.

3.4 Signal Processing

As has been discussed, EEG signals are inherently noisy. It is important, however; to define exactly what "noise" means in the case of these signals. While a traditional signal-to-noise ratio (SNR) applies to the electrical noise on the line, SNR, in the case of EEG, also applies to the background processes of the brain. The "noise" of stray electrons for brain activity need to also be filtered out and characterized for BCI development. A convenient definition for the SNR of a measurable, desired,

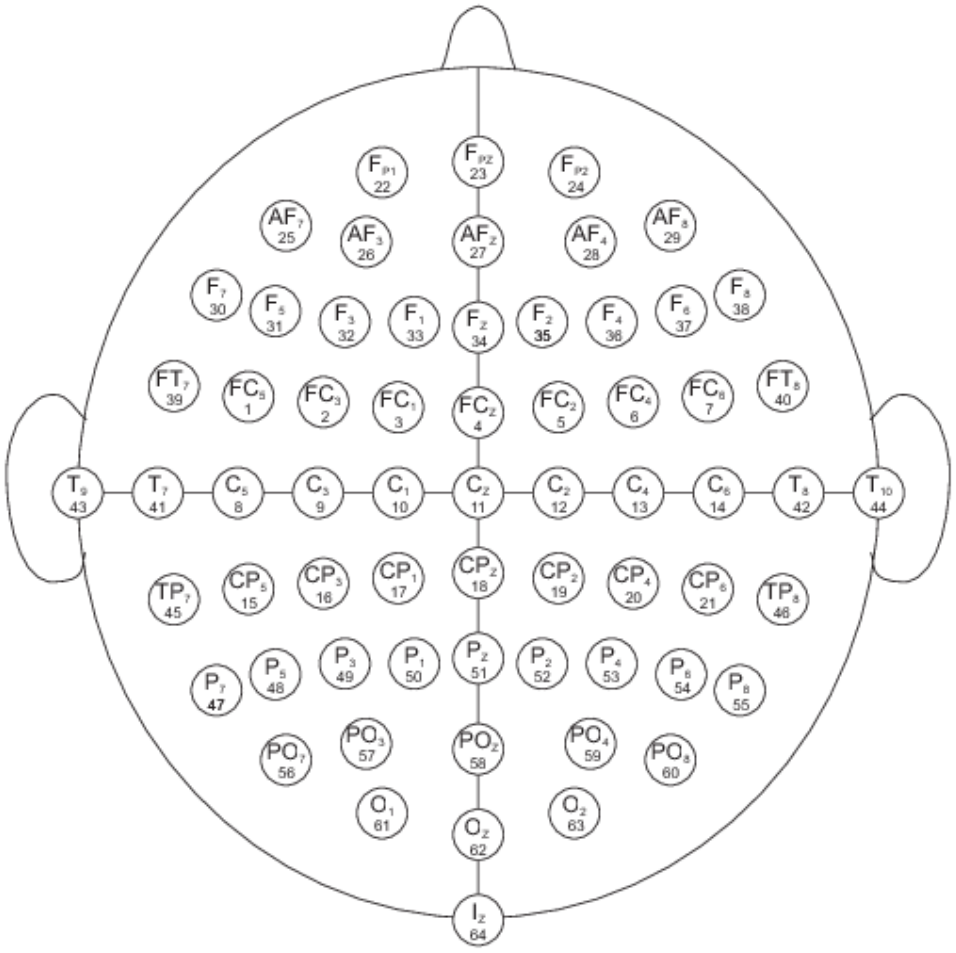


Figure 3.3: The International 10-10 Labeling System for EEG Recording, Excluding Electrodes Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10 [6].

characteristic is to compare the signal with and without that stimuli present. If the noise and signal are uncorrelated, which is taken care of by ICA, it is then possible to determine the signal without noise through a linear subtraction to equate the SNR. Since EEG requires many different recordings and trials to characterize a signal with machine learning, this process is slightly more involved due to the need of averaging to take all of the trials into account. Since the result is an average of the SNR over time, this is an approximation, due to the variance in physical setup, electrode placement, and other factors that change from recording to recording. For any N number of signals the SNR can be calculated by

$$SNR = \frac{\sum_{n=1}^{N} Signal_n - Rest_n}{N}$$

where *Signal* represents the active signal time window and *Rest* represents the EEG during inactive time windows, effectively the noise that should be removed to reproduce the pure active signal.

3.5 Robot Operating System

Rather than building, maintaining, repairing, and validating software solutions on many physical systems in a lab, it is a better use of resources to simulate the behaviors of the hardware on freely available software platforms. Currently, there are quite a few options that give researchers and hobbyists flexibility in choosing which simulator works best for each project [42]. A common factor in most of these simulators; however, is the abstraction between the hardware and software layers. These middleware abstractions make the portability of the code easier in eventually moving to physical systems. Due to the open source nature of these tools most of the solutions are implemented on a UNIX based platform.

Two of the most widely used systems that are open source are the Robot Operating System (ROS) and Gazebo. ROS has the advantage of allowing users to help develop

the platform though packages and code repositories. Another strength ROS has is the ability to interface with other middleware and a variety of simulators. The core concept of ROS being that ROS should be able to read, interpret, and communicate between devices that largely have no concept of communicate protocols between them. In this application that means that MATLAB need only be able to read in the signals from a device and output a generalized command of what the output should be. This can be as generic as an integer that has been classified as being an event.

Rather than having to develop all of these tools for a standalone platform the focus is on integration with a variety of other tools to allow researchers to use the tools they are most comfortable with and still have the end product work. This is due to the fact ROS is designed to be a partially real-time simulator so it is, theoretically, easy to transition from the simulated environment to the real world. Gazebo fits into this picture by integrating with ROS to introduce a 3D simulated environment. This allows a researcher to build an environment, program a system, and test the system in that environment.

CHAPTER 4: MATLAB TOOLS FOR BCI PROCESSING

4.1 Introduction

MATLAB is a natural toolbox for BCI development; primarily because having the EEG data stored in arrays where each row corresponds with an electrode's signal is intuitive and easily manipulated without touching the other signals in the array. In addition to just using MATLAB, many institutes and businesses have contributed to the body of knowledge through add-ons; the two most widely used ones being EEGLAB and BCILAB from the Swartz Center for Computational Neuroscience. This chapter introduces a basic example on how these tools are used in MATLAB to generate a working BCI, based on previous work by the Swartz Center. It is important to note that this example is mostly presented in an idealized form for the sake of example and some nuances are omitted and will be discussed later.

4.2 Example of BCI Development in MATLAB

To start analyzing the data, it must be first loaded into the EEGLAB. EEGLAB abstracts the handling, ordering, and manipulating of the data arrays that contain the raw EEG data. This abstraction is handled mostly though a graphical user interface so the developer of the BCI need not be proficient in MATLAB. All of the manipulation and filtering of the raw EEG data will be performed in EEGLAB, while all of the generation of the BCI to be used for a control system is completed in BCILAB. Since the data come in the EDF+ format a special plug-in for EEGLAB must be used to import the data. The option for downloading plug-ins can be accessed through the GUI for EEGLAB, or from the EEGLAB website. It is import to note that the option to import data must be selected, no the option for importing a dataset. A dataset of

the modified EEG signal will be generated later for import into BCILAB. Once the correct data has been loaded an additional window will pop up; this window is shown in Figure 4.1

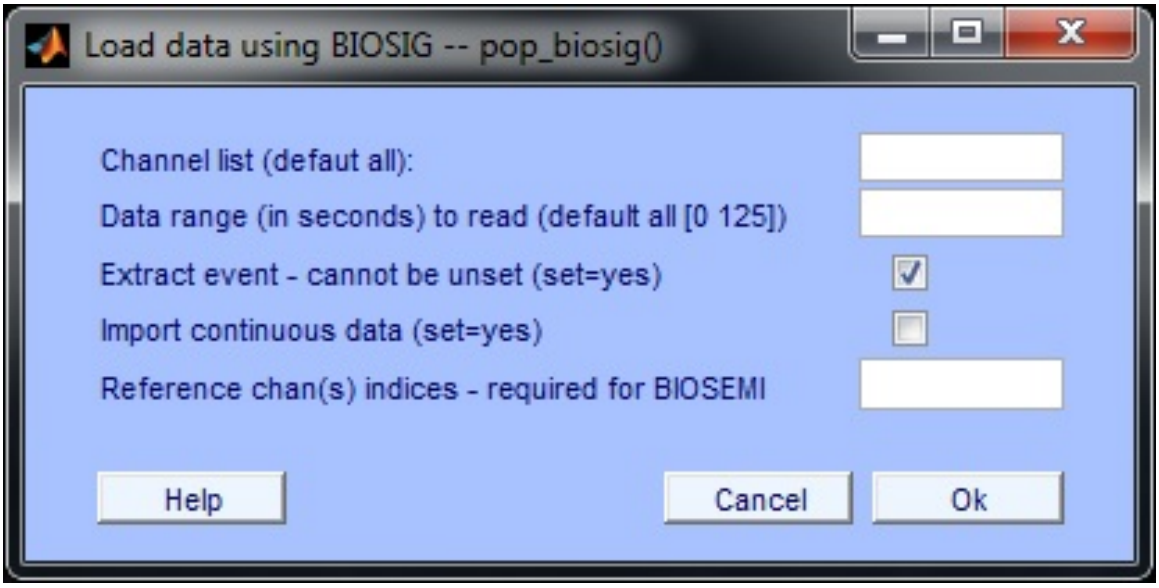


Figure 4.1: Menu to change the channels read by EEGLAB.

To look at the data all that needs to be done is to use the "Channel data (scroll)" option under the plot menu. The plot menu contains many useful functions that will be used later to Analise the results of the operations that will be performed on the data. For now, all that shows up is the raw data, as shown in Figure 4.2.

What is important to notice about the data is twofold; firstly, the signal is quite noisy and secondly, there are many more signals present than those that we care about. To remove the noise on the signal, it is as simple as passing the data through a filter, here a standard Butterworth bandpass filter. This can either be implemented in MATLAB or abstracted through the EEGLAB interface. The kind of filter that is implemented here depends quite heavily on the type of analysis that is desired on the end product. Since what is desired for creating this BCI based off of imagined movement, only those frequencies closely related to motor movements are important for the analysis. The beta waves that contain motor movements are located in the

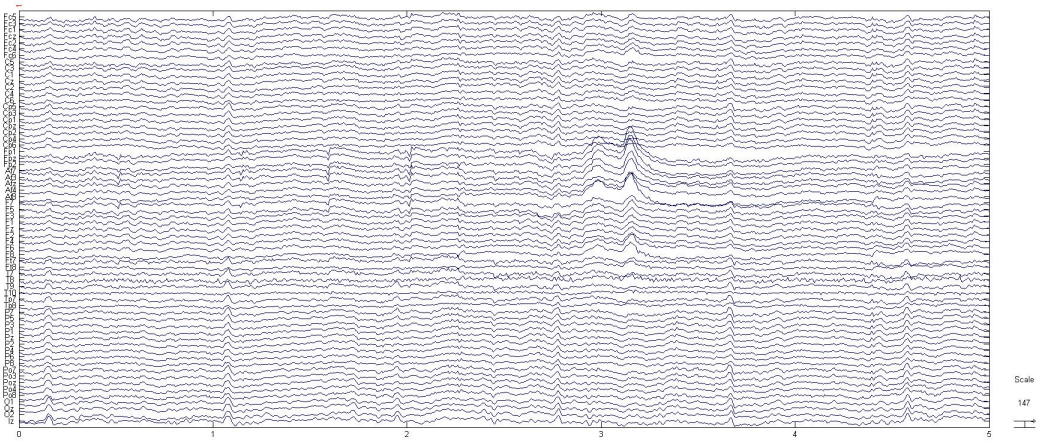


Figure 4.2: Raw unprocessed waveform of EEG recordings from the trial.

15Hz to 30Hz, roughly. By using a bandpass filter it is possible to isolate the beta waves that exist inside the signals. Implementing a bandpassfilter with at these frequencies the waveform now looks like Figure 4.3.

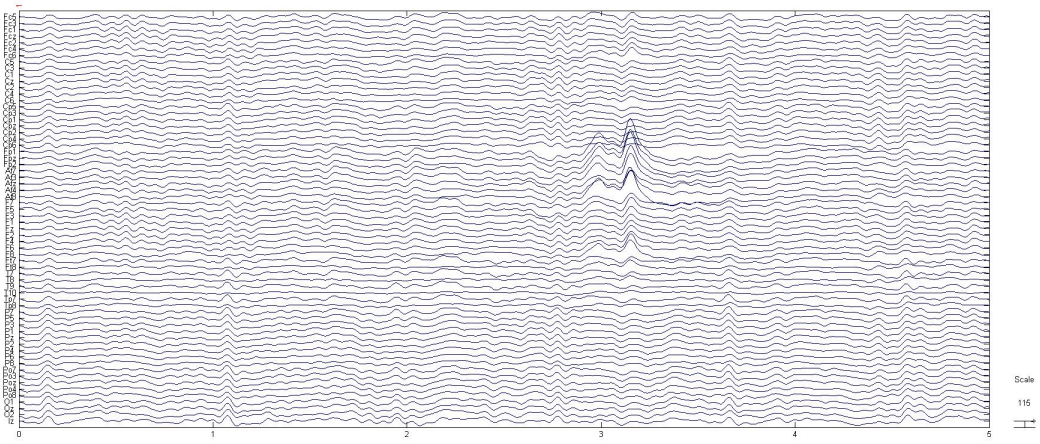


Figure 4.3: Filtered signal that removes noise on the signal.

As is evident in Figure 4.3 it is possible to see an example of the signal that we would like to extract and characterize. Now that the signal is filtered, the baseline

signal has been normalized and the spikes on the signal are much more prevalent. It is now much easier to see the activated signal and because of this it is easier to see the electrodes that are associated with the signal. This will help in the future in determining that a signal has occurred.

4.2.1 Event Duration

It is important to not only know when a visual target was presented to the participant, but to know for how long it was displayed on the screen. Since the participants were instructed to continue the activated motion for a long as the target was presented on the screen, knowing how long the target was displayed will help with selecting the pertinent data associated with each motion. The more precisely and consistently the associated data can be paired with the corresponding motion, the more precise and consistent the resulting model of the control system will be. The data for how long each tarted was displayed to the participant is embedded in the EDF+ files that contain the raw EEG data. EEGLAB can extract the data contained in the EDF+ file to annotate and display the duration of the presented target. By loading in some sample data from one of the subjects and choosing 'show event duration' from the 'Display' tab at the top of the graph the duration of the events is shown. From Figure 4.4 this duration is shown to be 4 seconds by looking at the X-axis of the graph, which is represented in seconds. How long the event is present in the data was dependent on the dataset used in the study. Generally, in gathering EEG recordings from an individual this will be how long the subject was presented with a certain stimuli, in this study an arrow on the screen. In EDF+ file recordings where, and how long, these events are contained is stored in an array in the file.

4.2.2 Data Signals Associated with Motor Movement

To find which of the signals relate to motor movement, ICA must be performed. An ICA is a special case of a blind source separation. A famous case of blind source

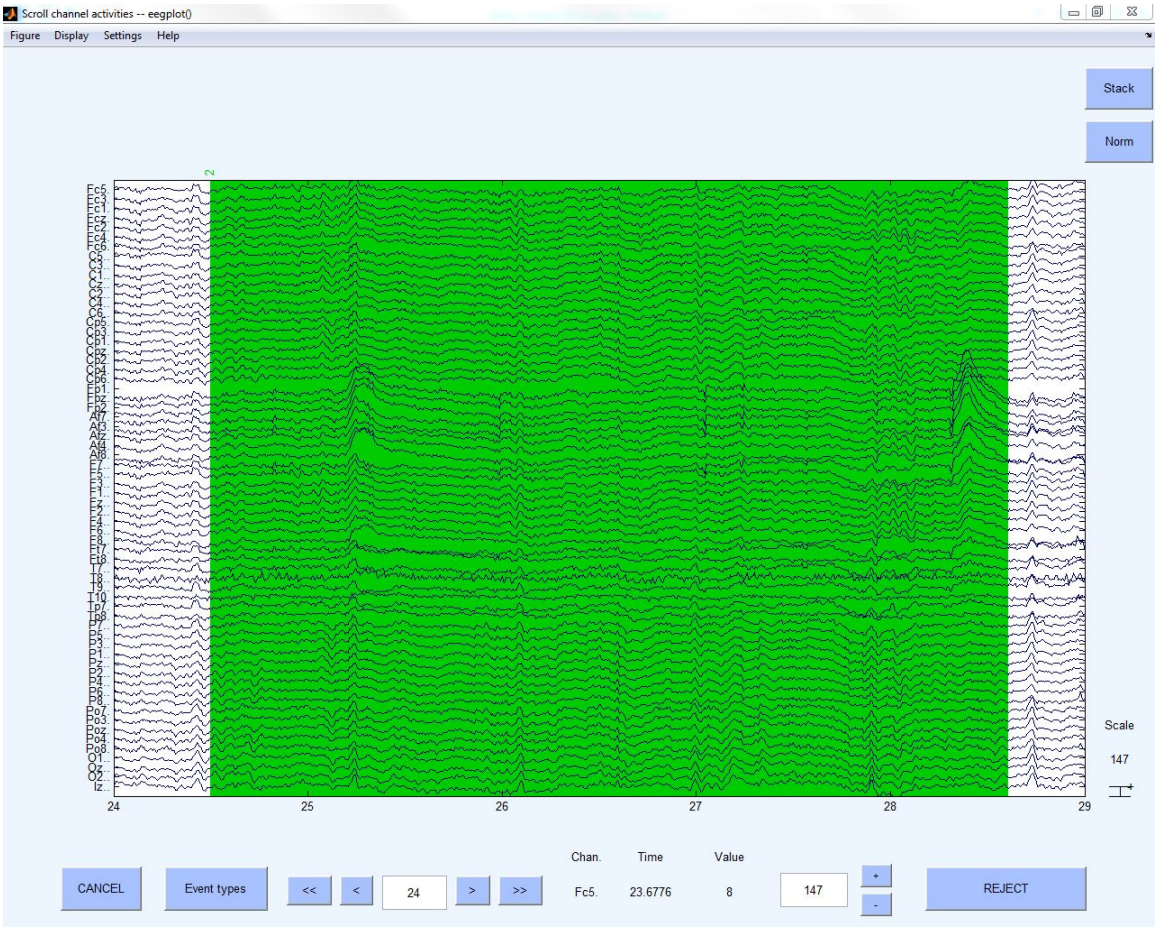


Figure 4.4: Green area highlighting duration of a typical target that is presented to a subject. This target was present from 24.5 - 28.5 seconds of the trial.

separation is a problem called the "dinner party problem." This problem is defined as being at a dinner party and being able to separate the audio signal of one individual attending the party. One can use the principal of blind source separation to solve this problem. The key to solving this problem is also the key to separating the signals on the EEG cap; there must be as many measurement devices as there are signals. In the case of the dinner party problem there must be at least two measurement devices, one for the background noise and one capturing the desired signal of the person talking. This is simplification of the method involved, but the same principals apply for separating out the independent signals from all of the compiled EEG noise.

If an ICA were to not be performed on the data it would not be known how to weight

all of the individual electrodes due to the signals represented on each electrode would be the additive contributions of the entire signal rather than the independent component. An ICA on the entire cap would be an unnecessarily complex computation though. It is possible to simplify this computation by only evaluating the electrodes that are known to be associated with what is being evaluated, motor movement. In essence the ICA will show which signals are the most closely associated with motor movements. At this step of the development process it is not an issue of confirming which movement took place, but that a movement took place at all. It is important to note that since we already know what type of signal we are looking for in the data, a motor movement, and the position of the motor cortex, which controls motor movement in individuals, is already known from previous research [43], as shown in Figure 4.5, the ICA can be simplified somewhat, thereby reducing computation time and complexity.

Since the signals from the brain are dipole signals, meaning that they have both a magnitude and direction, the results of the ICA provide enlightenment as to the radiation of the signal patterns also. Even though some of the signals were discounted because it was previously known that those signals would not contribute to the desired outcome, after the ICA other electrodes may be discounted. This is mainly done by manual inspection of the component maps, one of which is shown in Figure 4.6. Only the components with present or localized activation regions were chosen. It is important to notice that the labeling of the electrodes has changed from the initial 10-10 Standard shown in Figure 3.3. This is because of the channels that were removed for the ICA. The updated numbering format is shown in Figure 4.6.

It is important to note that this procedure needs to be repeated for every trial that is needed to create the BCI for the control. The figures that are present were derived from following the listed procedure from the trials detecting left hand and right hand movements. These, along with the baseline trial, will be used in the machine learning

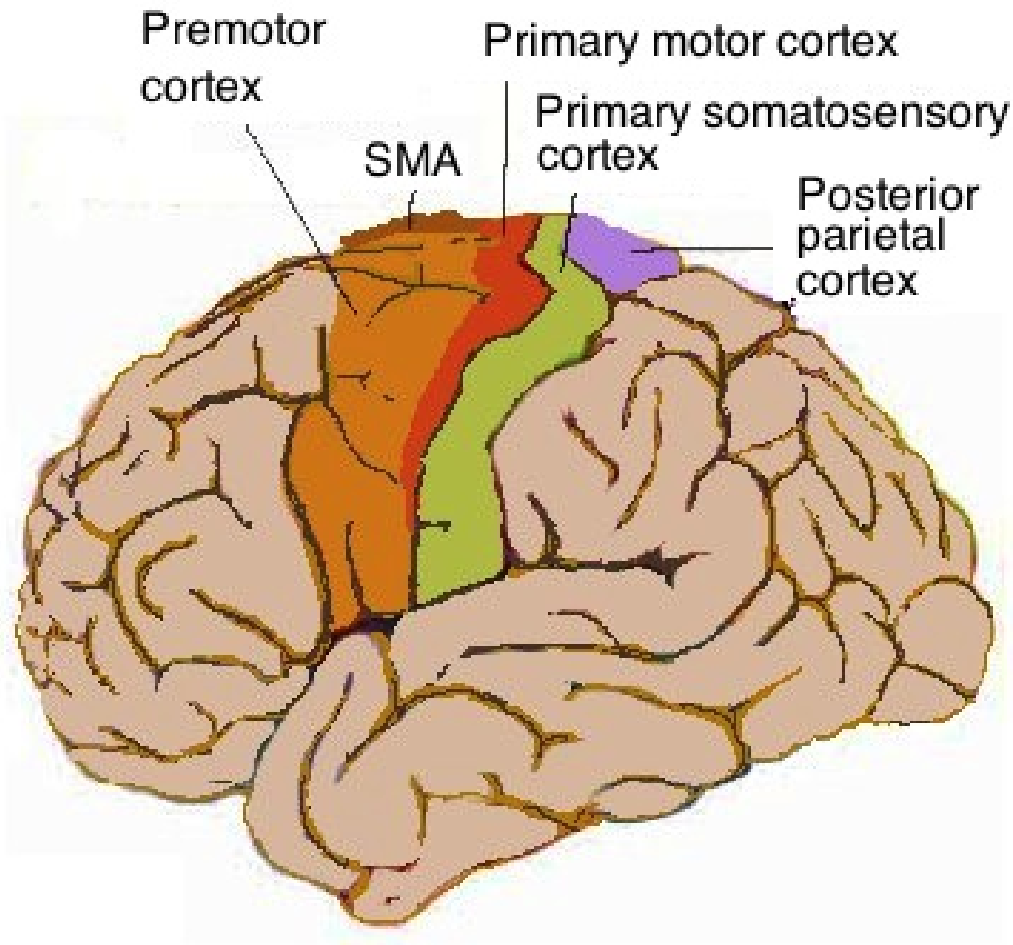


Figure 4.5: Picture showing the relevant sections of the brain for motor movements.

algorithm to train a model for the imagined movement signals. In order to now translate to BCILAB to create the BCI, all the signals that have been processed in the above mannor need to be saved as a dataset in EEGLAB. This creates a ".set" file that is imported into BCILAB.

4.2.3 BCILAB

After downloading BCILAB and adding the folder to the MATLAB pathway run the bcilab.m file to start BCILAB. The first time that the file runs it will take a while to build all of the configuration files. Once BCILAB start a window will pop up that has all of the options for BCILAB present in a graphical user interface (GUI) format. This window is shown in Figure 4.7. As an aside, it is possible to write scrips to run

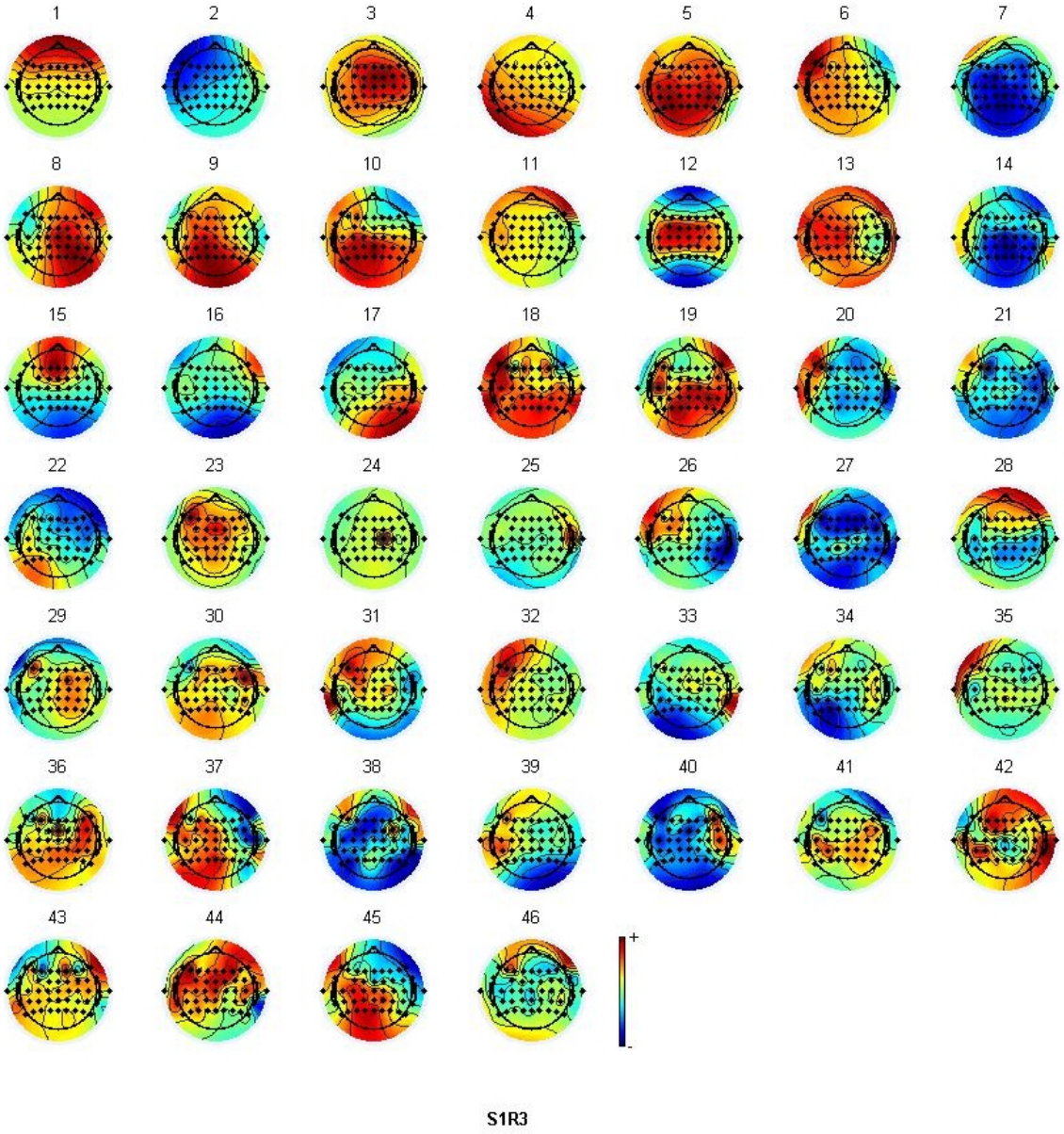


Figure 4.6: Weighted ICA of one participant's trials of hand movements. This is effectively a heat graph, with higher intensity areas colored in red, and lower intensity areas transitioning to blue. This was generated with the help of the EEGLAB plugin for MATLAB.

and compile all of the options presented here, but for this paper the GUI format will be shown for clarity of the procedure.

Using the "Data Source" option on the menu and selecting the "Load Recording(s)" option it is possible to now load the ".set" file that was saved by the earlier

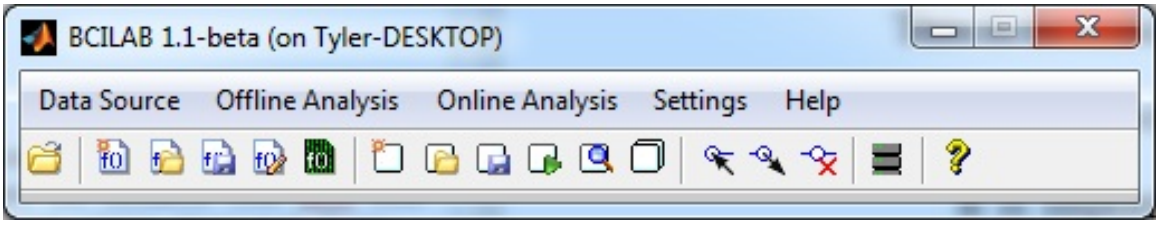


Figure 4.7: Main menu of the BCILAB plugin for MATLAB.

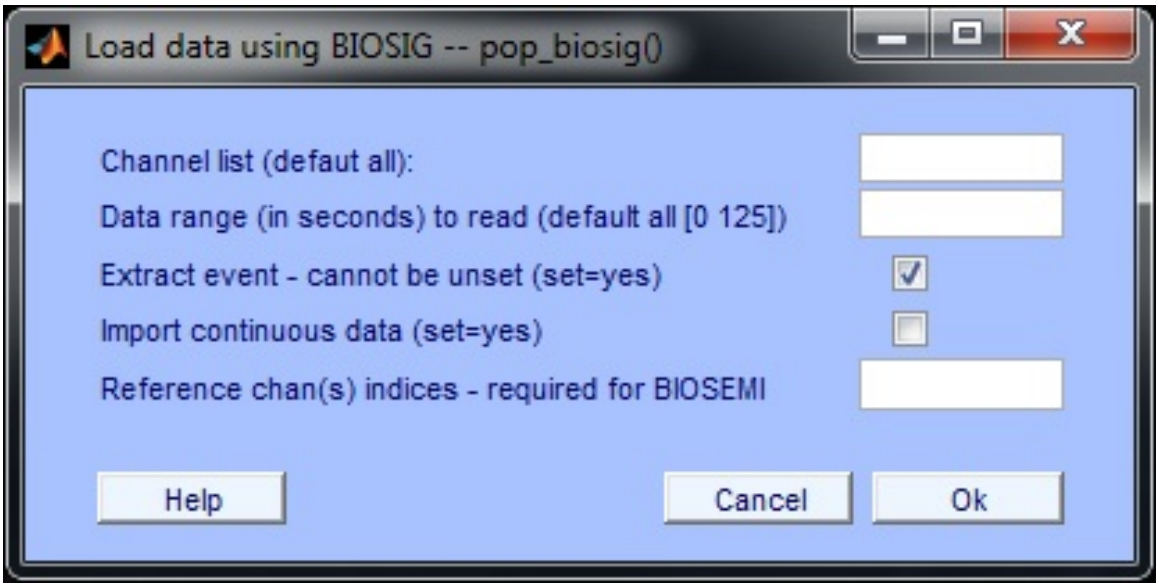


Figure 4.8: Menu and options for loading the dataset file into BCILAB.

procedure. The file that needs to be loaded first is one of the trials that used actual muscle movements. It is very important to load one of the trials that used actual movement because it will be used for training a model to find the signals in the imagined movement files. For now, the file containing the data for the left and right hand movements are loaded. Once the correct file is selected the window shown in Figure 4.8 will pop up. From this window it is possible to select the channels that will be analyzed. All of the channels from Figure 4.6 were used since the signal has already been processed to the degree that all of the pertinent signals are present, so this field was left blank. The data can also be truncated based on time, but since we use the whole of the recording, this is also left blank. The other options are for other features of BCILAB that are not used in this development, but are useful for

developing a BCI from other means, such as live recordings or importing data in other formats.

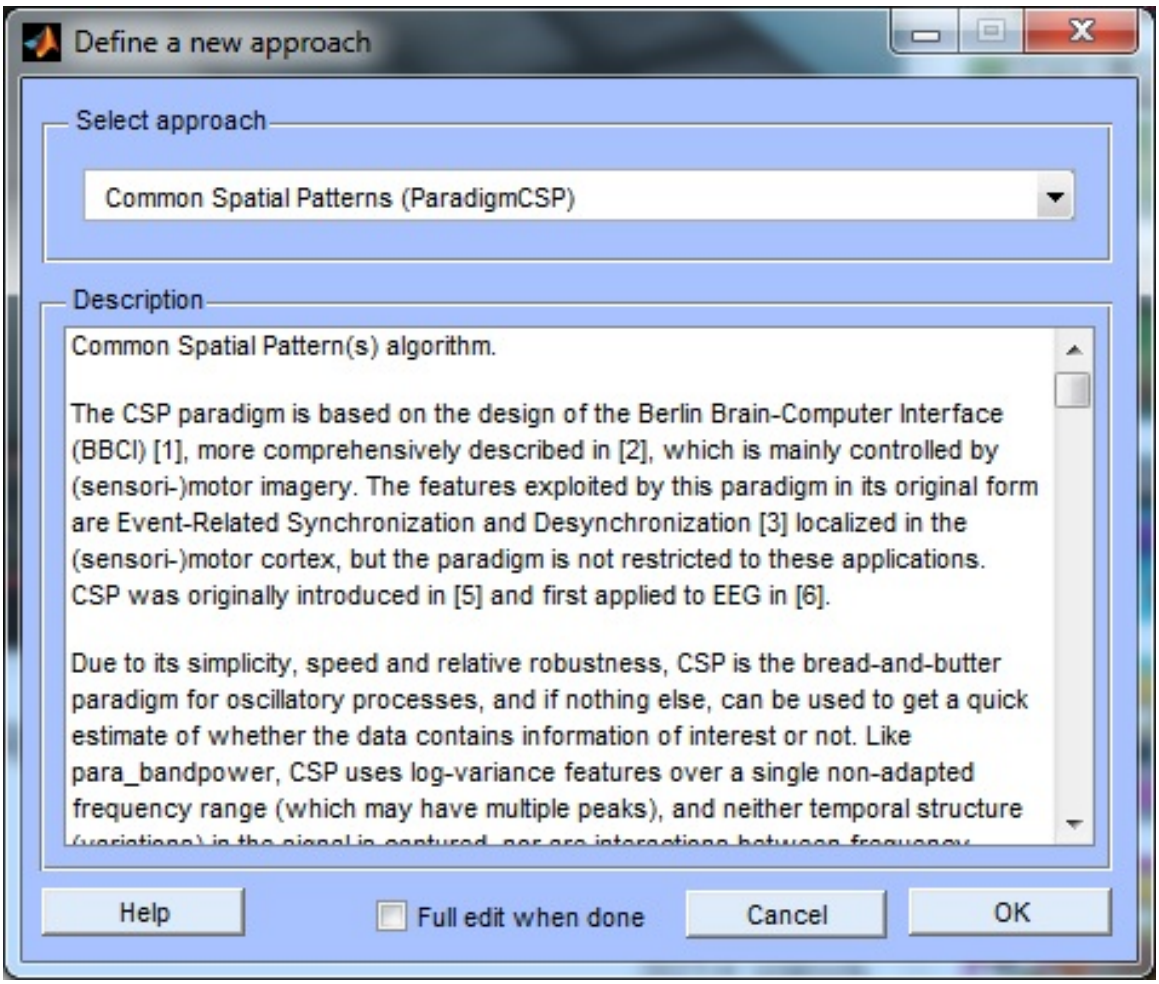


Figure 4.9: Window for choosing the approach for the model. Common spatial pattern approach shown.

Now that the data has been loaded into BCILAB it is time to start developing the BCI that will be used for the control system. By selecting "New Approach" under the "Offline Analysis" from the main menu, it is possible to design an approach for our BCI. The approach that is selected from the menu on the window that pops up will define what kind of BCI that is developed. There are many options here for different kinds of BCI processing that can occur. The windowed-means approach was shown earlier, but what is desired now is an approach called common spatial patterns (CSP).

A brief description of each approach shows as it is selected, and one such description is shown in Figure 4.9. As the approach states, CSP is used for oscillatory processes localized in the motor cortex. This is desirable for the BCI what we want to create since it has to do with motor movement of the hands and feet. As is stated in the description, CSP is a very common and robust approach so we will examine on how to refine this approach later to tailor it more closely to what we are developing.

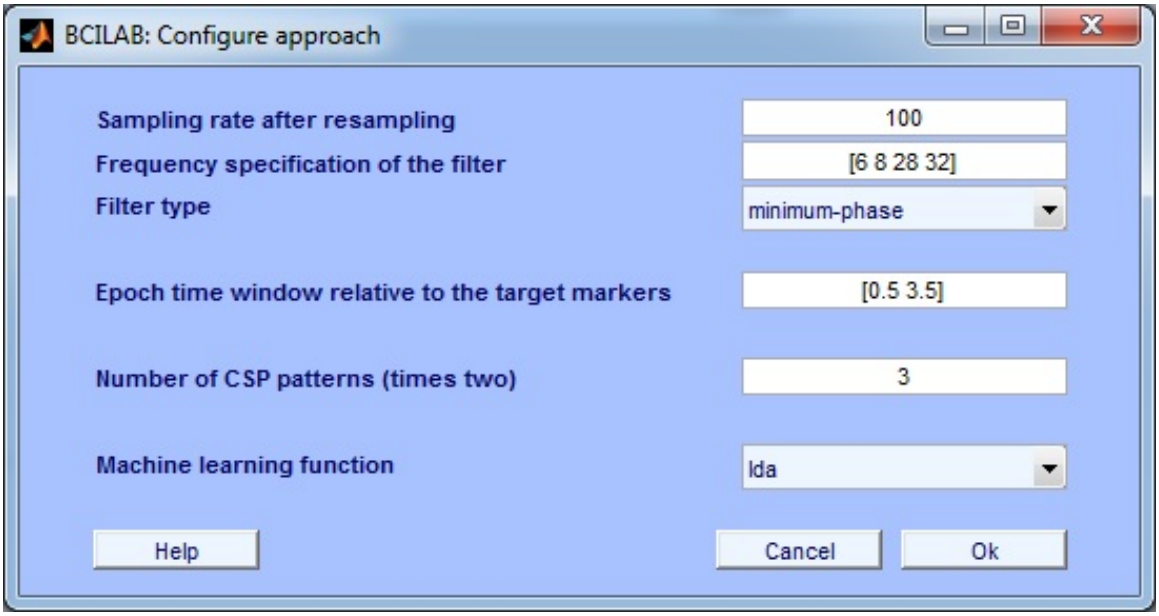


Figure 4.10: Option for CSP approach.

After selecting the CSP approach another window, shown in Figure 4.10 will pop up. This window contains a lot of properties of the CSP that can be changed to refine the approach. The CSP approach filters the data to look for the embedded oscillatory processes, in our case beta waves for motor control, so the default approach configurations of the frequency specifications of the filter can be left alone. Also it is possible to narrow the time window of the epoch for each signal. As was shown earlier the signal window of each signal that was presented on the screen for the trial was 4 seconds. It is desirable to trim this slightly to account for reaction times of the physical signals, so half of a second is trimmed from each side of each epoch. The final important function that can be chosen is the machine learning function for

the approach. It is possible to choose from a lot of different functions for the CSP approach for fitting the data for a model. The default linear discriminant analysis (LDA) function is chosen here to highlight how accurate the default settings are. It is worth changing this to different functions and checking the results to find the best model, but, as will be shown, the default LDA function can be an accurate learning function. It is important to note here that it is possible to over characterize the approach for the model. This occurs when the approach is too closely configured to the data from actual motor movement that it is difficult to distinguish an imagined movement. Over training and fitting the approach to the model will result in it being difficult for the BCI to correctly identify the imagined signals due to them being too dissimilar from the tightly trained model.

Figure 4.11 shows the window that opens after selecting to train a new model through the main menu. The window already loads in the last approach made and the last data loaded into memory. For the target field it is important to select the field that contains the data for the markers for the signals. Due to how the data that is being used was saved this field is labeled "type". Also, since the data, as shown earlier, contain markers for rest along with the two markers for activation (either hand or foot movement depending on the trial being examined) it is only desirable to look at the markers that contain activations. Only the markers 1 and 2 contain the data needed to direct the model to the appropriate signal times. Since the duration was trimmed earlier, the model will calibrate using the data contained in these signals with half a second delay and cutoff. The rest of the options can be left to the defaults for now, but they may be changed in the future to refine the calibration data.

By selecting the correct options shown in Figure 4.11 the results shown in Figure 4.12 are obtained. Looking at this data it is shown that after five iterations of machine learning the model is around 95.4% accurate for the model. This means that with the values that were chosen for the model were very good for allowing the model

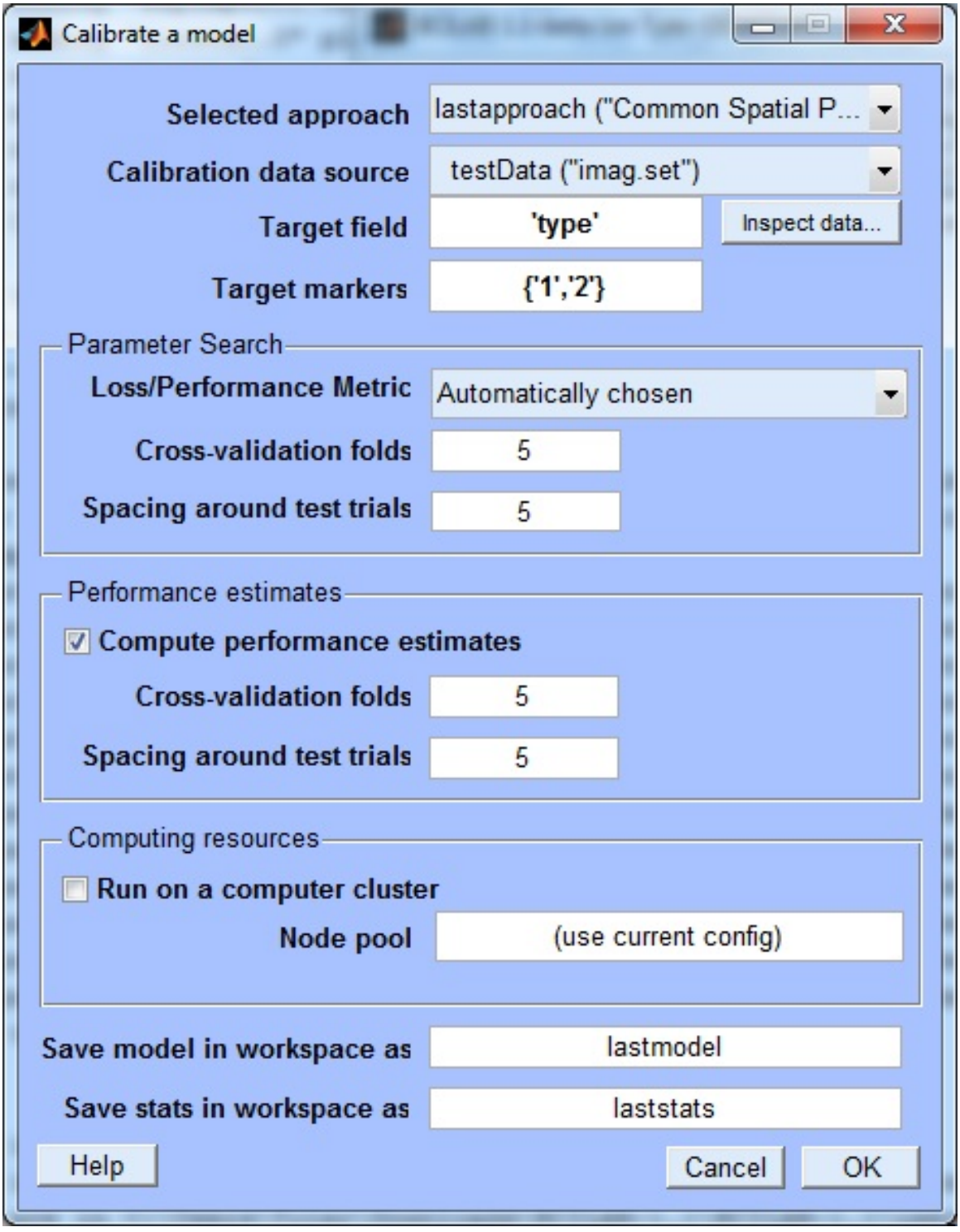
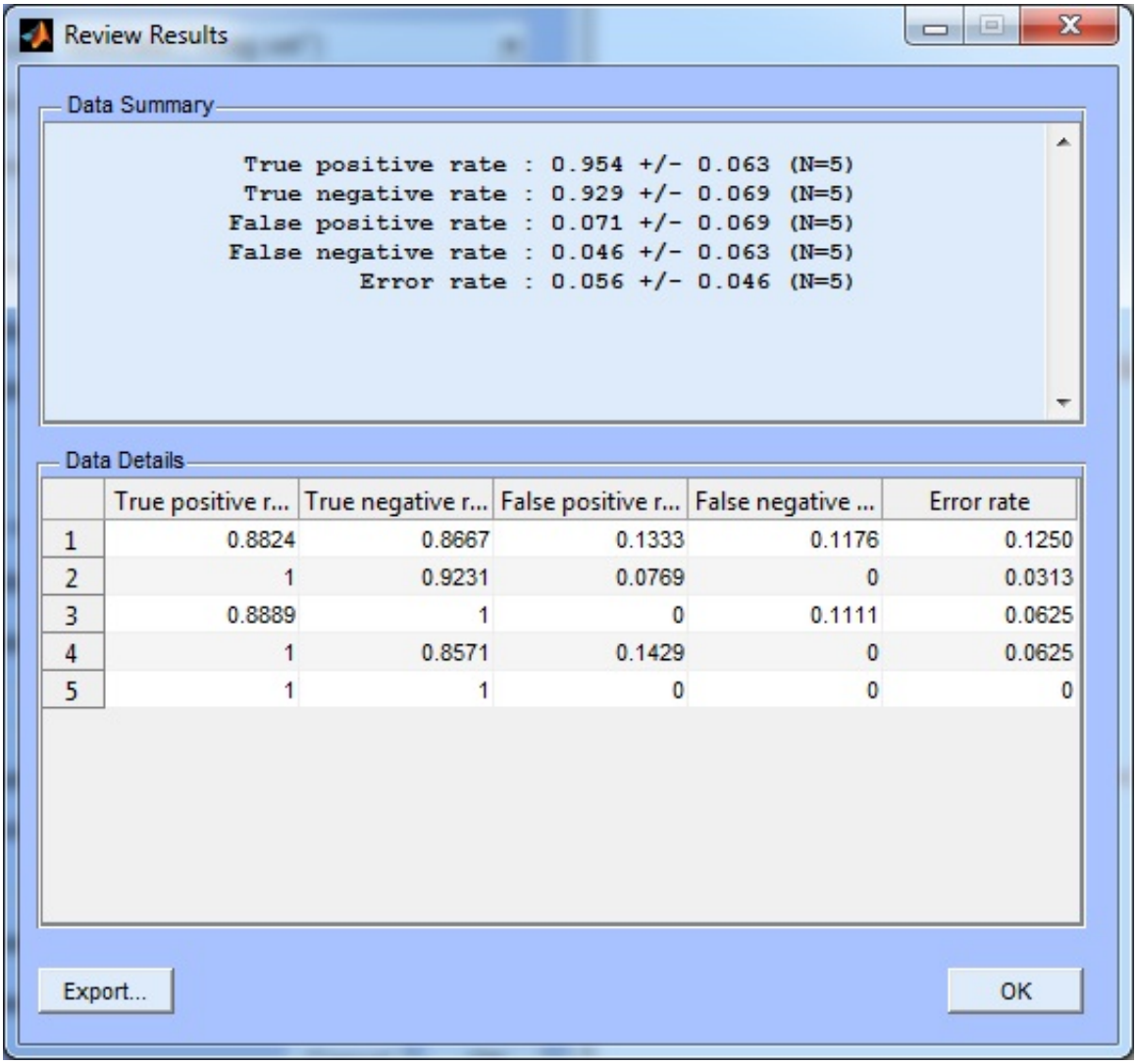


Figure 4.11: Calibration options for generating a model.

to predict that a move has taken place. Again, it is important to note that these steps must be taken with every dataset in order to create predictive models for each



	True positive r...	True negative r...	False positive r...	False negative ...	Error rate
1	0.8824	0.8667	0.1333	0.1176	0.1250
2	1	0.9231	0.0769	0	0.0313
3	0.8889	1	0	0.1111	0.0625
4	1	0.8571	0.1429	0	0.0625
5	1	1	0	0	0

Figure 4.12: Results from the default parameters of the CSP model approach.

of the differing moves.

Now that the model has been created, it is time to fit it to the data. By selecting "Apply model to data" from the main menu, it is possible to see how well this model that was developed will predict and detect that the correct movement has taken place. First though, the dataset that contains the imagined movements must be loaded into memory using the steps listed beforehand. Once this data is loaded it is available for selection in applying the model to the data. Once the process has ran the results shown in Figure 4.13 are obtained. From these results we can see that after only one

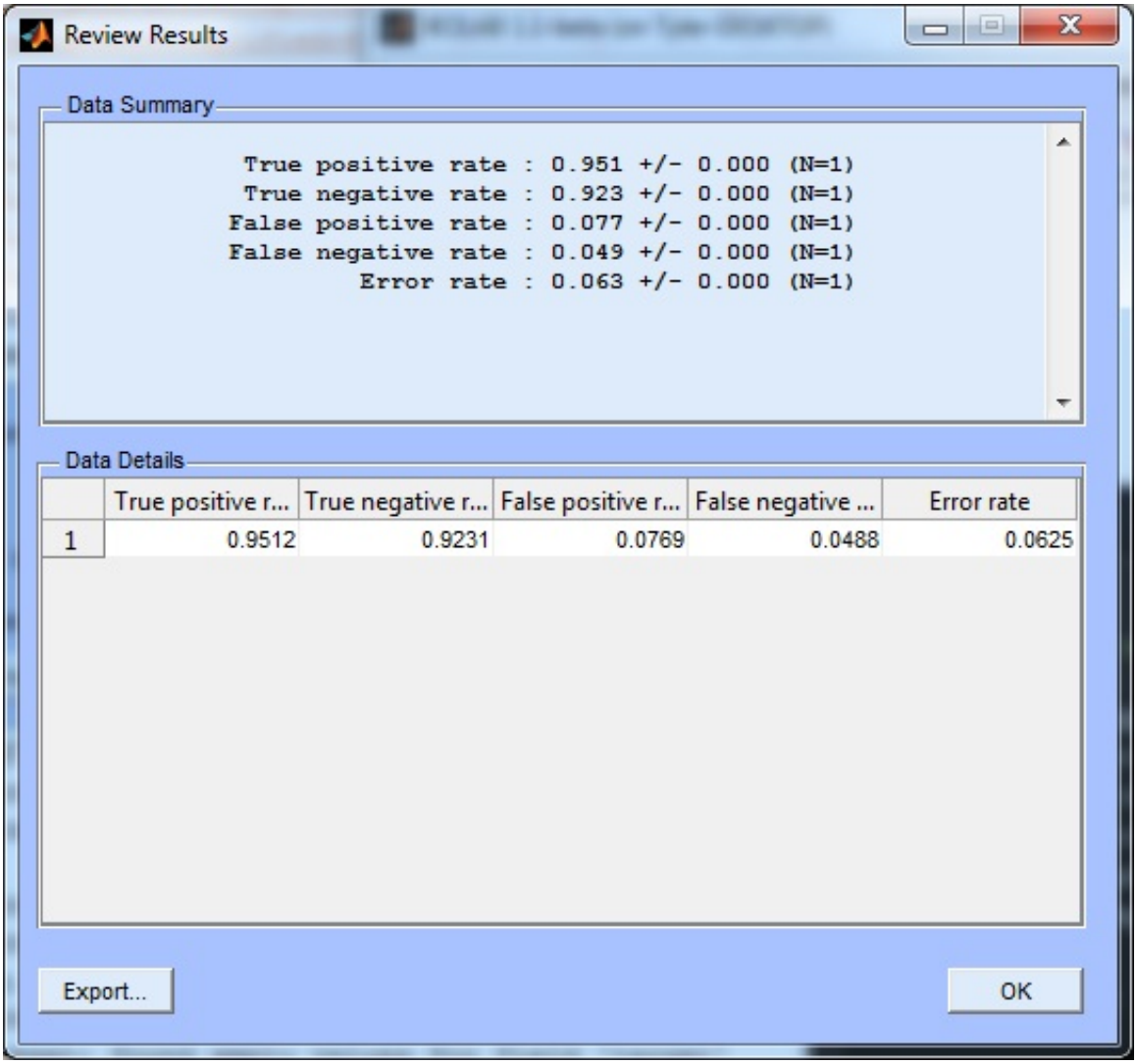


Figure 4.13: Results from using the generated CSP model on the imagined movement data.

pass of the imagined movement data, the model that we generated through CSP was successfully able to predict the correct movement 95% of the time. This is therefore shown to be a very good model. Again, these are the results for the data containing only the hand movements; although similar results were obtained using the other datasets.

While this model that was generated is already very good, there is room for improvement. Let us take a look as to how changing the approach for the model calibration can change the results on the same data.

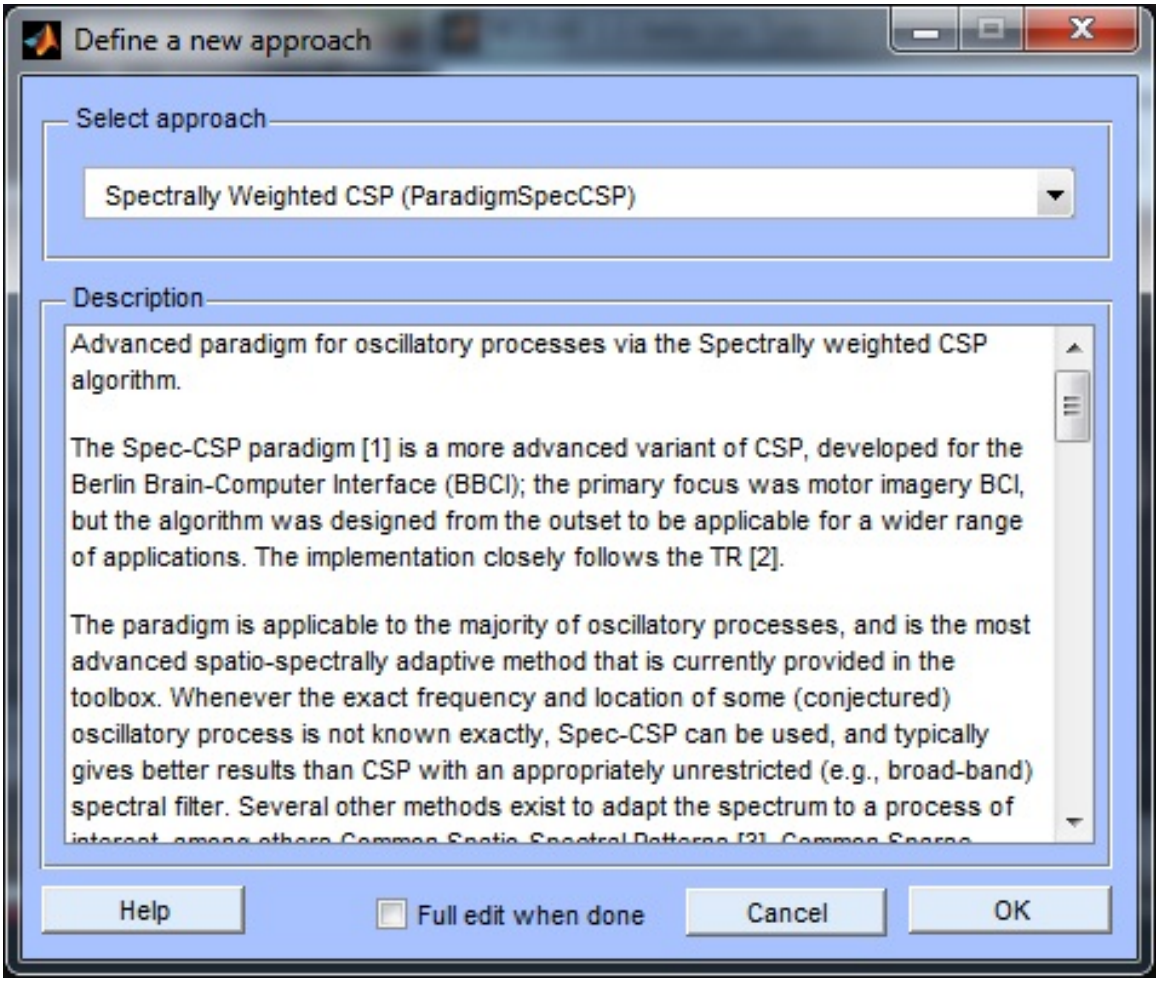


Figure 4.14: Approach of spectrally weighted CSP designed specifically for imagined movements.

The approach that is chosen this time is a special variant of the CSP approach called the spectrally weighted CSP. This is a more advanced form of the CSP that takes into account the weight of each electrode in the machine learning stages. This means that the ICA that was performed earlier will be used to spectrally weight each signal for importance. This approach was originally designed for motor imagery BCI in mind, as stated in the description in Figure 4.14, so it is especially suited to develop our desired BCI.

By looking at the configurations for the spectrally weighted CSP in Figure 4.15, we can see that the frequency specification of the filter has been changed slightly, and

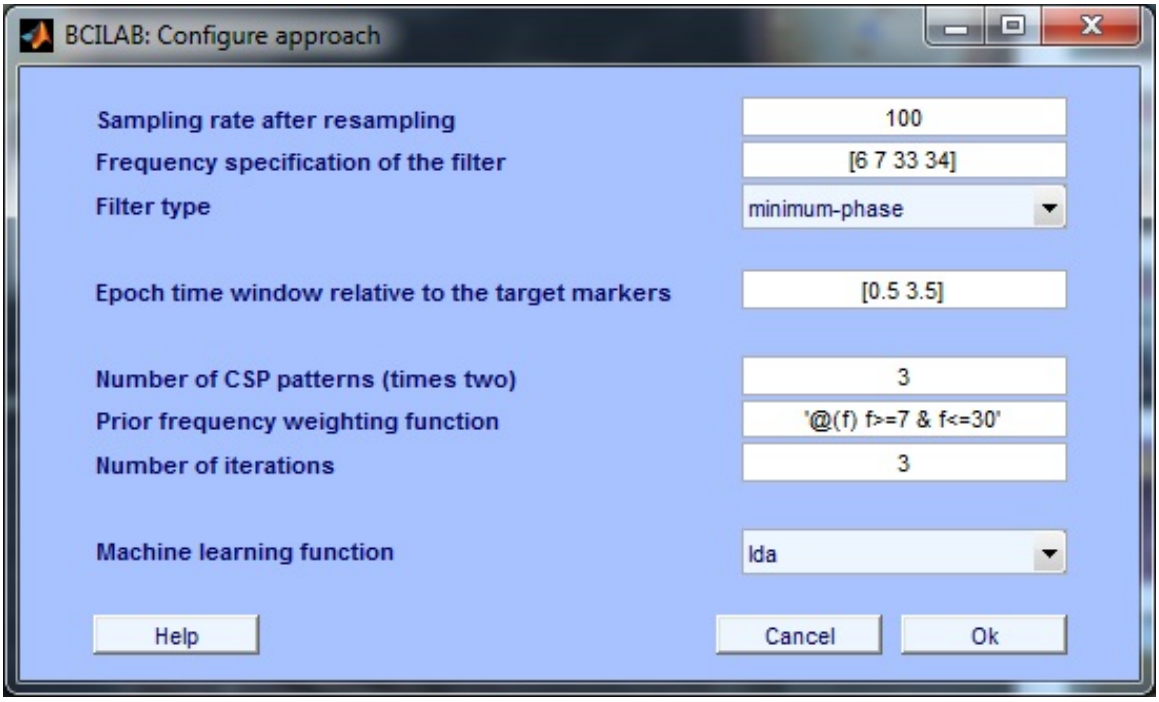
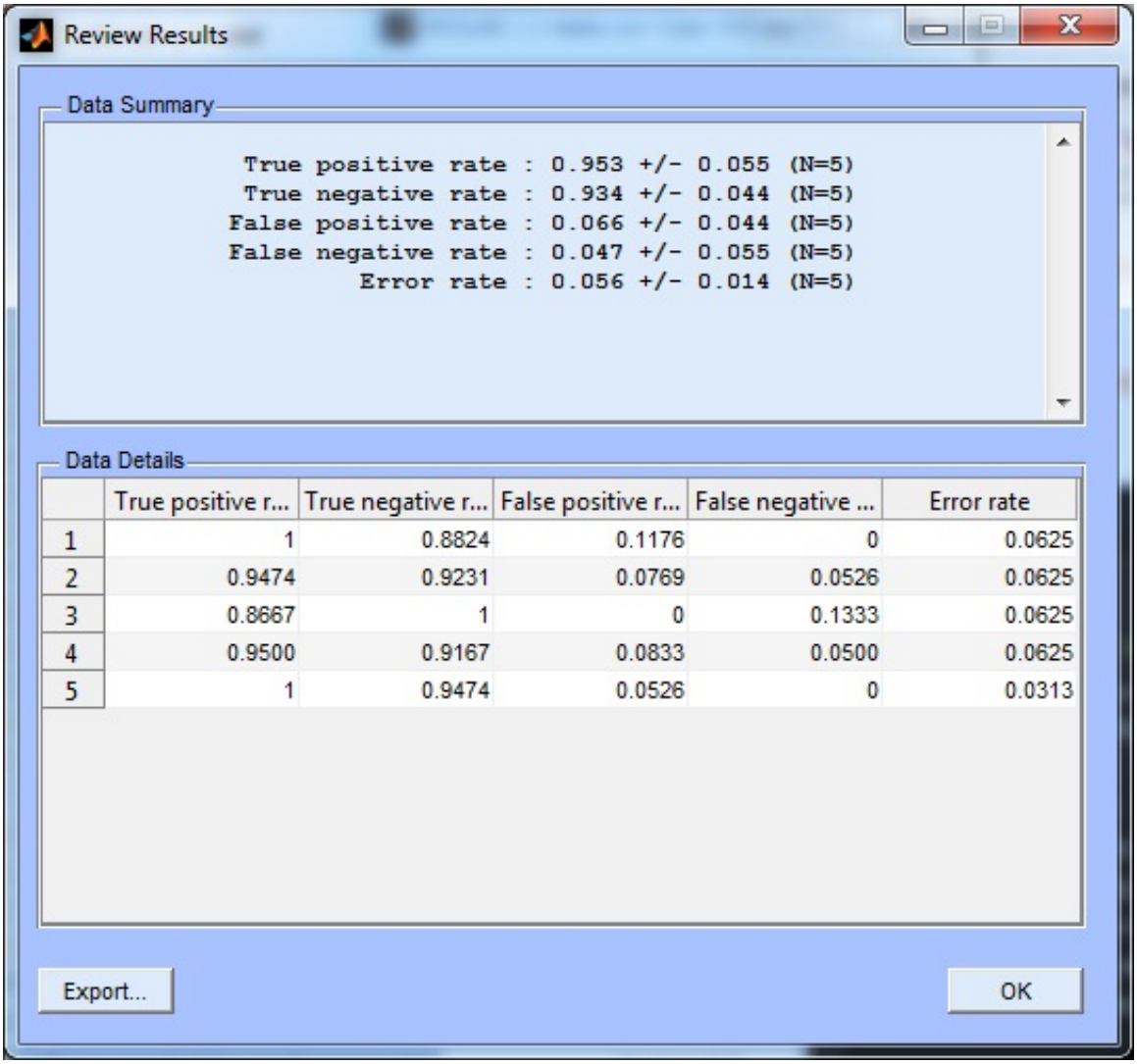


Figure 4.15: Configurations for the spectrally weighted CSP approach.

there is an option to modify the prior frequency weighting for the function. To keep the same weights as were originally generated by the ICA, this should remain as a default. Everything else was kept as the default values to more accurately show how just changing the approach to one more suited to the desired task will change the accuracy of the model on the data.

After training the model with the calibration data as was shown earlier, the results from Figure 4.16 are calculated. At first glance these results seem quite similar to the data obtained, only slight variations occur. This is due to the fact that the same machine learning approach, LDA, was used to generate both of the models from the same data. The variations that exist occur because of the difference in approach, so the fact that the results of the machine learning on the calibration data are so close is due to the function used.

The main variation of the results comes when the model that we trained is used to analyze the imagined movement dataset. Since the spectrally weighted CSP was



	True positive r...	True negative r...	False positive r...	False negative ...	Error rate
1	1	0.8824	0.1176	0	0.0625
2	0.9474	0.9231	0.0769	0.0526	0.0625
3	0.8667	1	0	0.1333	0.0625
4	0.9500	0.9167	0.0833	0.0500	0.0625
5	1	0.9474	0.0526	0	0.0313

Figure 4.16: Results of the generated spectrally weighted CSP on the calibration data.

designed with this purpose in mind, it is safe to expect that this model, even will all other settings remaining the same, to perform better than just the standard CSP model. Looking at the results in Figure 4.17 it is shown that this is easily the case. Looking at these results, the model that was generated successfully predicted the correct movement 98.8% of the time. Not only that, but the error rate and false positives of signal detection were also greatly reduced. This shows that by choosing the correct paradigm for the model that the accuracy of a BCI can be greatly improved.

In order to put this model into a more visual and intuitive perspective it is possible

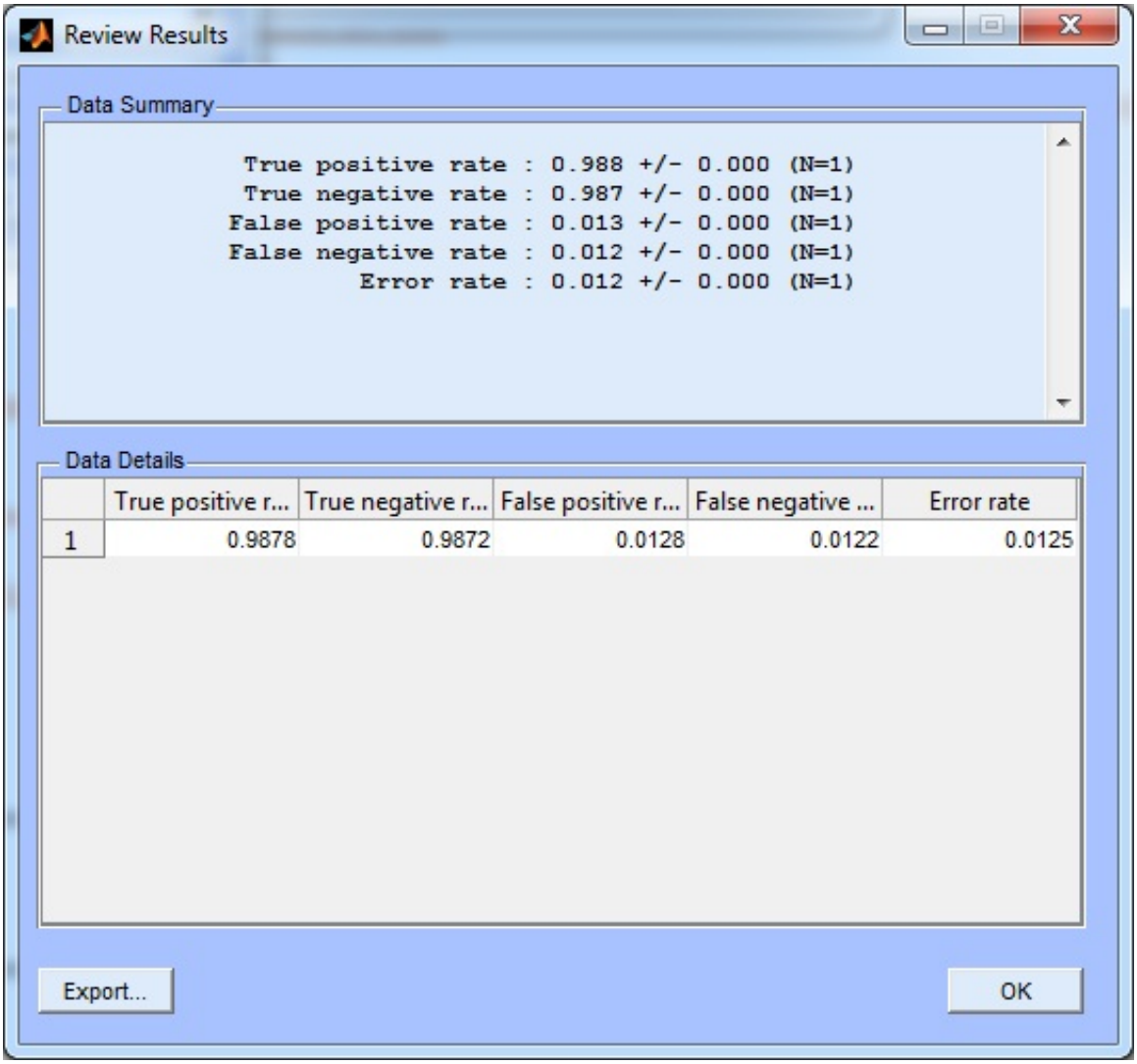


Figure 4.17: Final performance results of the spectrally weighted CSP model on imagined movement.

to use the "visualize model" option under the BCILAB menu. Doing so yields the results in Figure 4.18. What this result lets us see is the patterns that arise from the model and data. There are 6 patterns represented here and as we can see they all radiate from around the motor cortex region. This is to be expected since the dataset contains motor movements.

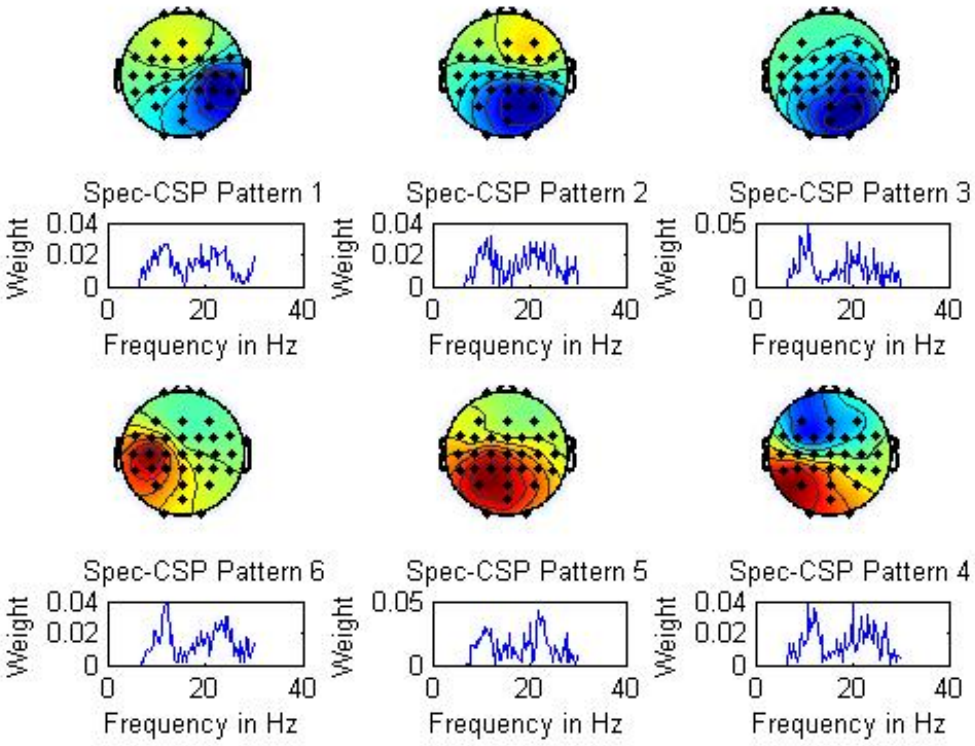


Figure 4.18: Visualization of spectrally weighted CSP model generated and the contributions of each frequency for each pattern that emerged.

4.2.4 Online Analysis

Even though the data and analysis up to this point has been completed purely on offline data, it is possible though BCILAB to simulate the performance of the model on the data stream as though it was an online signal. Once the model has been trained up to this point and the imagined data loaded into memory it is possible to select "read input from dataset" though the online analysis menu. Once this has been selected one must also select to output the data to MATLAB visualization though the same menu. A window with two bars will pop up. The two bars on this window will move up and down to simulate the model sweeping the data in real time and making a decision on if a signal is presented and which one is currently being shown.

CHAPTER 5: HARDWARE SYSTEMS

5.1 HAPTIX

The Defense Advanced Research Projects Agency (DARPA) is currently funding the Hand Proprioception & Touch Interfaces (HAPTIX) program for robotic limb replacements. The program is designed to "develop new science and technology to achieve closed-loop control of dexterous mechatronic prostheses that will provide amputees with prosthetic limb systems that feel and function like natural limbs." [44] Unlike most robotic limbs of the past, HAPTIX is centered on closing the loop and providing feedback to the user. So far, this research has partnered with Mobius Bionics LLC to develop the Life Under Kinetic Evolution (LUKE) system for arm and hand replacement [10].

The Open Source Robotics Foundation (OSRF) was tasked with simulating the prosthetic hand in Gazebo and building test environments with interfaces for development. Figure 5.1 shows what the current result of that research looks like.

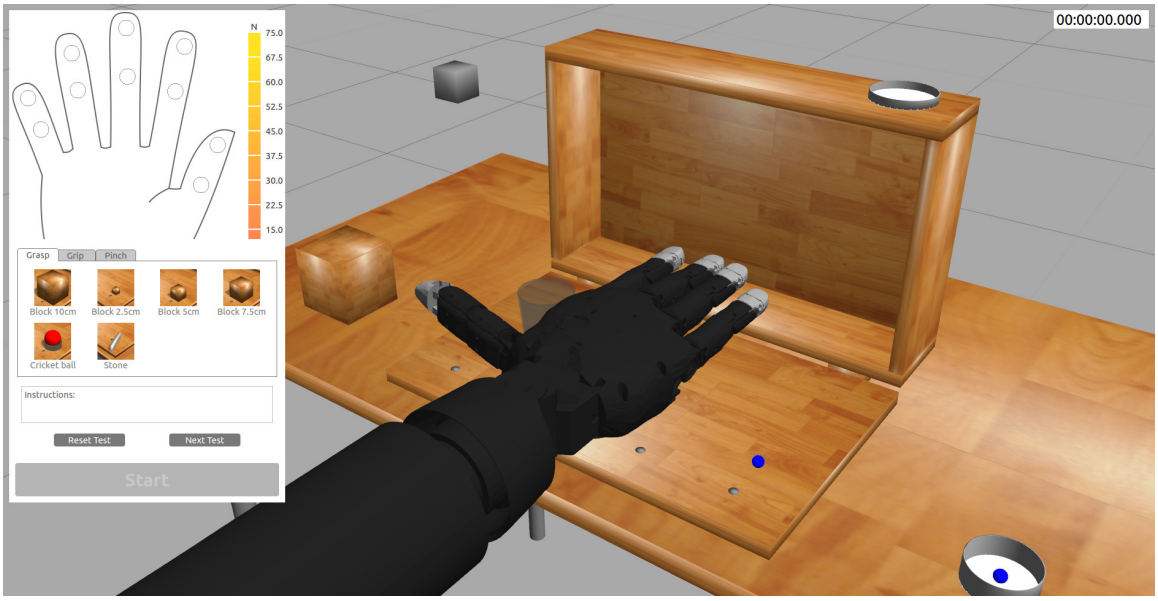


Figure 5.1: Example simulation environment of the DARPA HAPTIX robotic arm. This simulation includes a graphical feedback of the touch sensors on each hand as well as a list of the objects in the scene that can be manipulated [7].

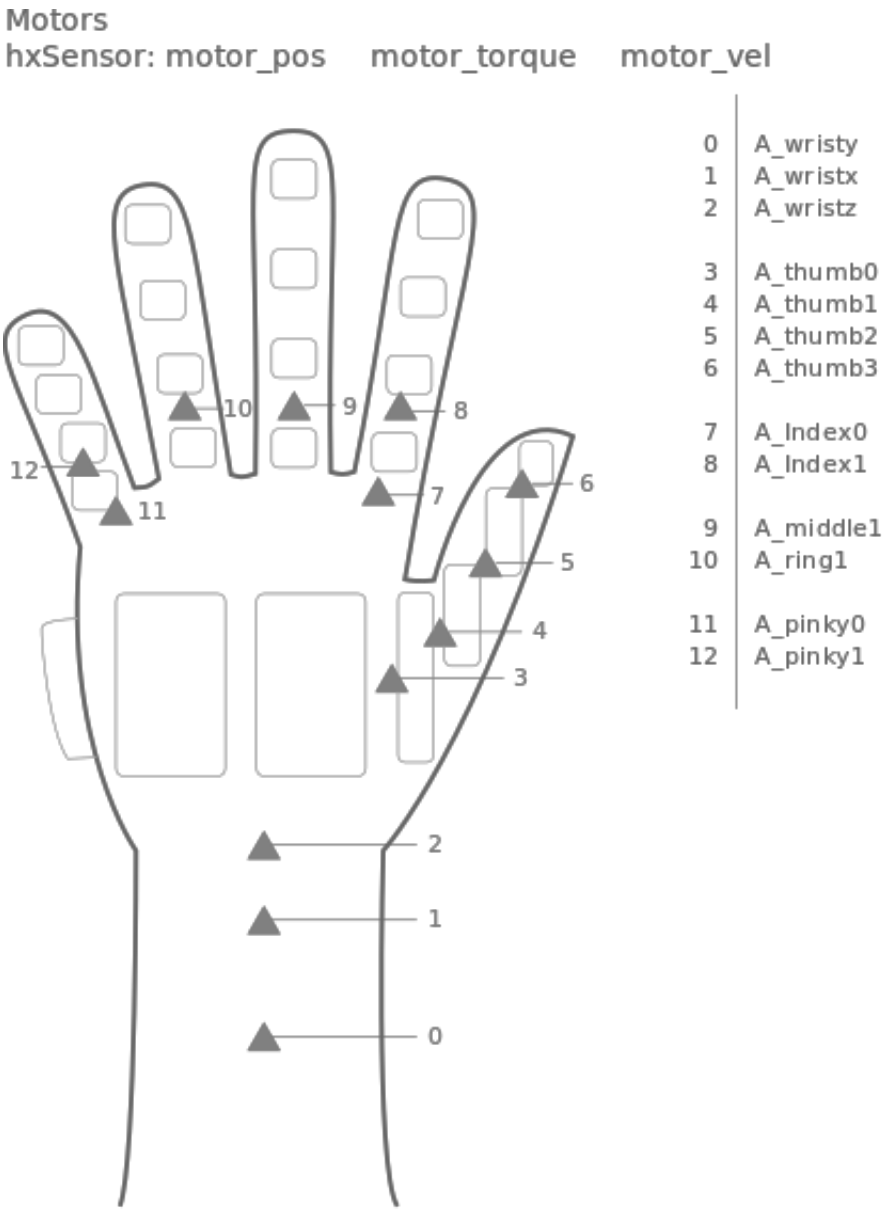


Figure 5.2: Schematic describing the location of the motors in the hand. The numbers on the diagram correspond to the index position of the array in the C API of each motor. Each motor provides feedback on its position, torque, and velocity [7].

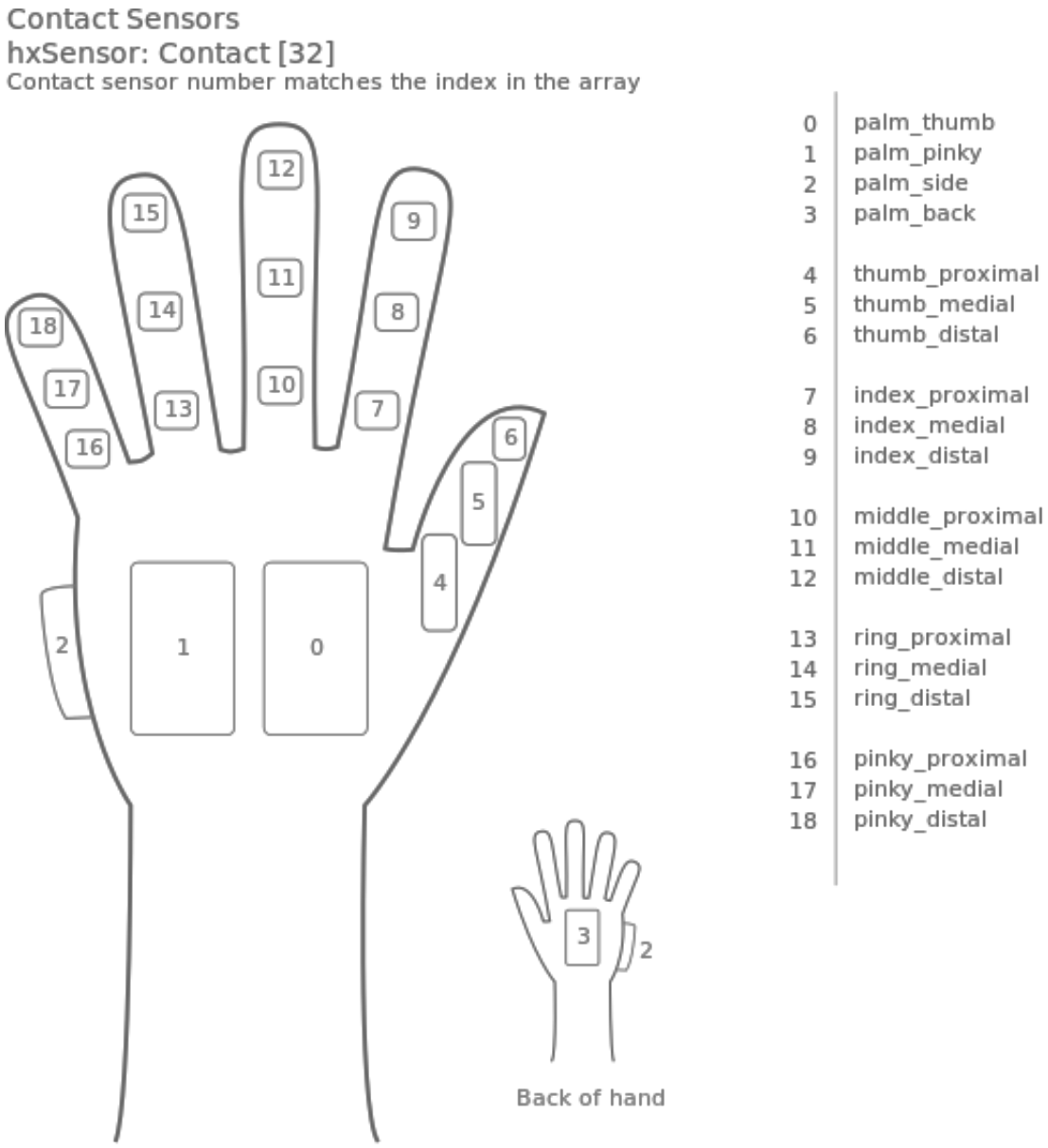


Figure 5.3: Location of contact sensors of the DARPA HAPTIX hand. The numbers on the diagram correspond to the index position of the array containing the sensors. There is also another array that contains the IMU data (located under the fingernails) to be used as additional sensor feedback [7].

5.2 InMoov

The InMoov is a humanoid robot designed by French sculptor Gaël Langevin [45]. The InMoov project started in January of 2012 as the first open source prosthetic hand; since then the project has expanded and evolved to include the entire upper

body. The entire shell of the robot can be printed with a 3D printer with a build size of only 12x12x12cm print area. Even though the InMoov is primarily supported through myrobotlab, an open source programming environment, code can also be developed to work with ROS. It is this open source nature, along with the, relative, ease of obtaining an end product, that has popularized the model and is why it was chosen for this research.

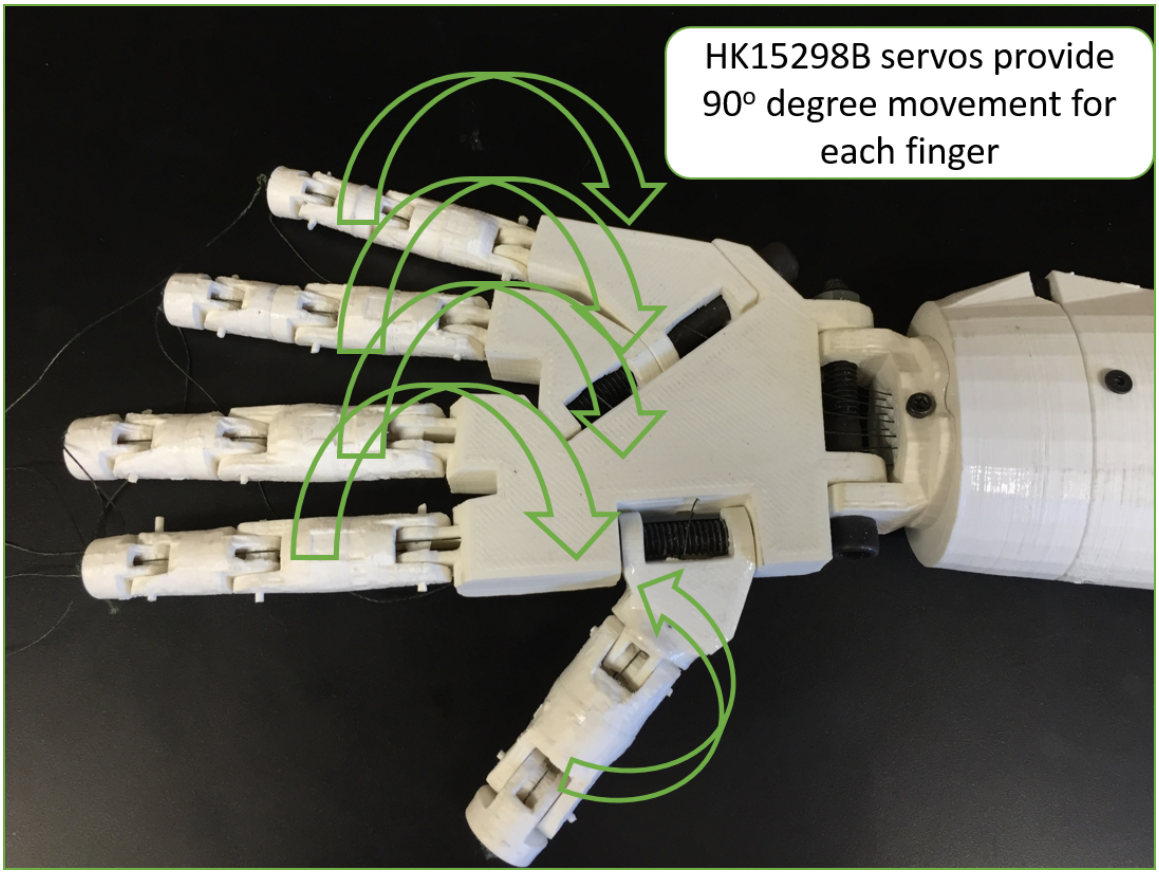


Figure 5.4: 3D printed had of the InMoov project that showcases a closeup of the mechanics of finger and joint location. Each servo, connected to each joint, provides 90 degrees of movement, enabling the hand to fully close.

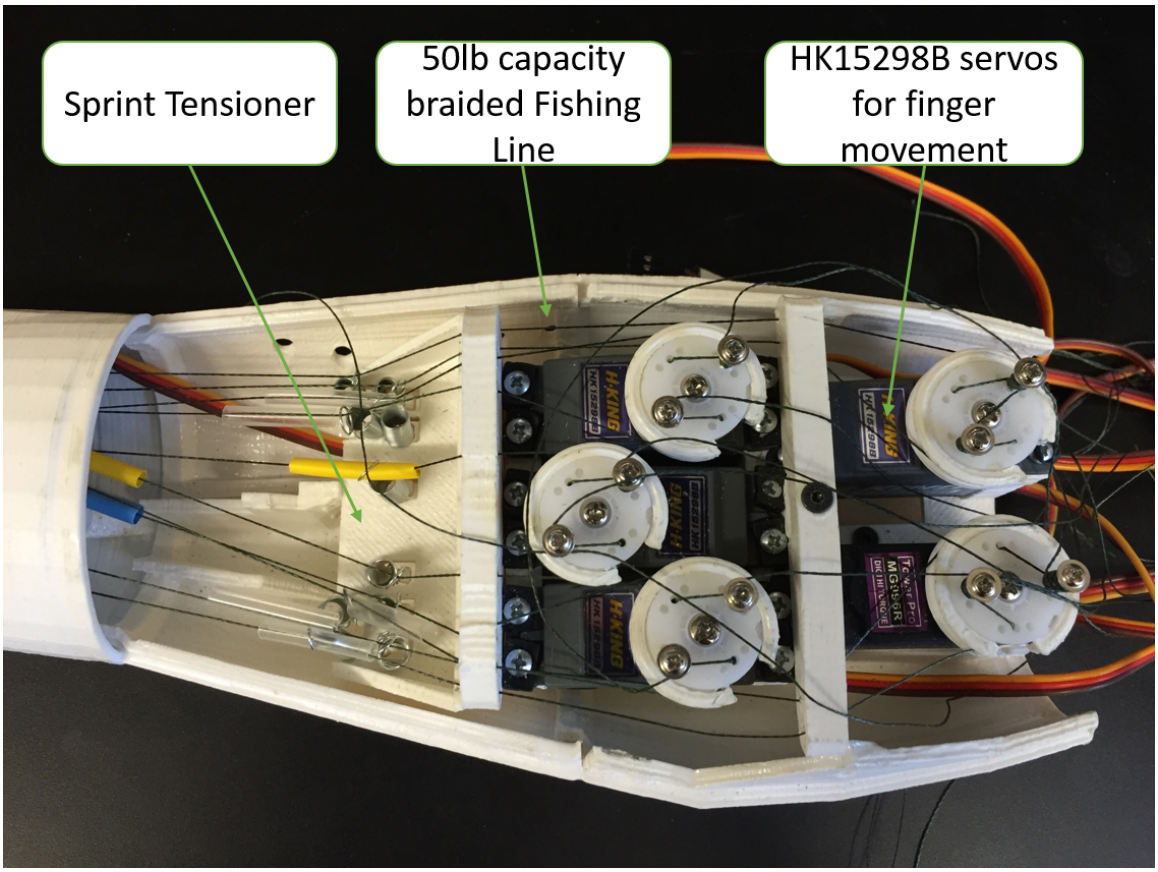


Figure 5.5: Forearm of the arm unit of the InMoov. While normally covered, this shows the tensor strings that connect up to the hand and the servos that pull to close the fingers.

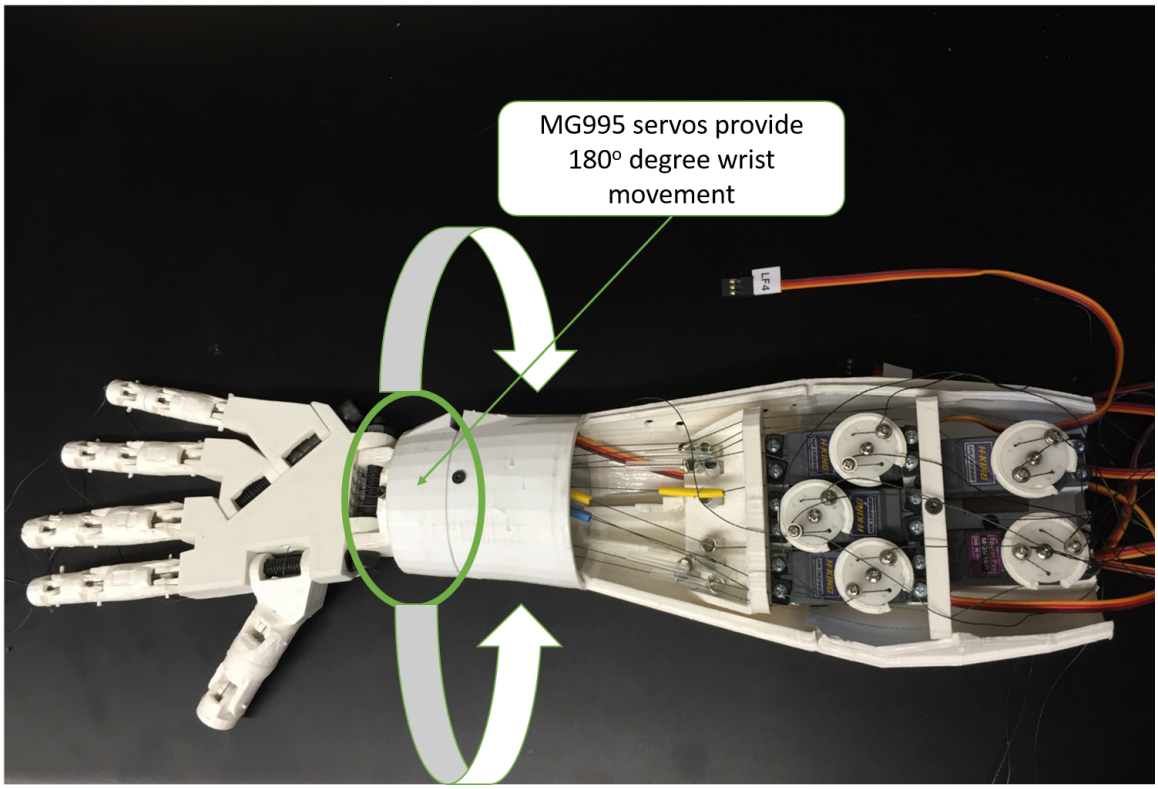


Figure 5.6: Highlight of the wrist movement capabilities of the InMoov arm. The wrist is capable of rotating 180 degrees, independent of the finger movement.

5.3 Gazebo

As mentioned previously, building and maintaining physical hardware can be an expensive endeavor. It is attractive, for researchers, to be able to test code and control schema on a variety of systems before committing to a purchase. Gazebo is a popular solution to this approach to design. Gazebo is a simulation and modeling tool used to design and test robots and artificial intelligence (AI) systems. The reason Gazebo has become so popular recently is due to it's free price tag and an active community. There are many freely available models of robotic prosthetics, both hand and foot, for testing.

To show how a simulated environment in Gazebo may be used to develop more than just prosthetics, Figure 5.7 shows the Turtlebot2 navigating an environment. It is easy to see from the figure that a variety of objects can be simulated in order to

develop any number of situations. Expanding this to the field of BCI development, prosthetics, harnesses, caps, and sensors can all also be modeled in order to give an overall picture of how a final design interacts when fully assembled and is moving.

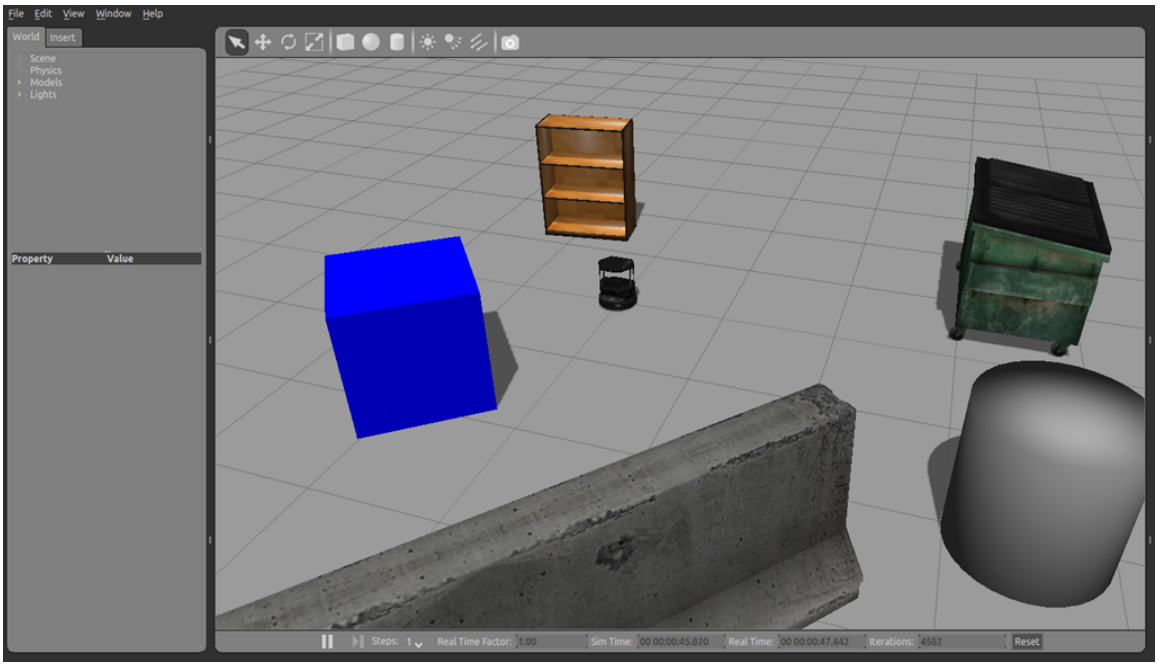


Figure 5.7: Turtlebot2 placed in a simulated environment, powered by Gazebo. Gazebo comes preloaded with example physics objects of a wide variety that are already detailed and properly scaled for the simulator.

Gazebo is compatible with both ROS and MATLAB, so it easily fits in with the tools discussed so far. ROS handles the communication between MATLAB and Gazebo, fulfilling its role as the information handling middleware [46]. Due to the prolific virtual machine (VM) usage with ROS, operating with Gazebo and MATLAB is only a matter of IP address passing. Figure 5.8 shows a typical flow of IP addressing for such a setup.

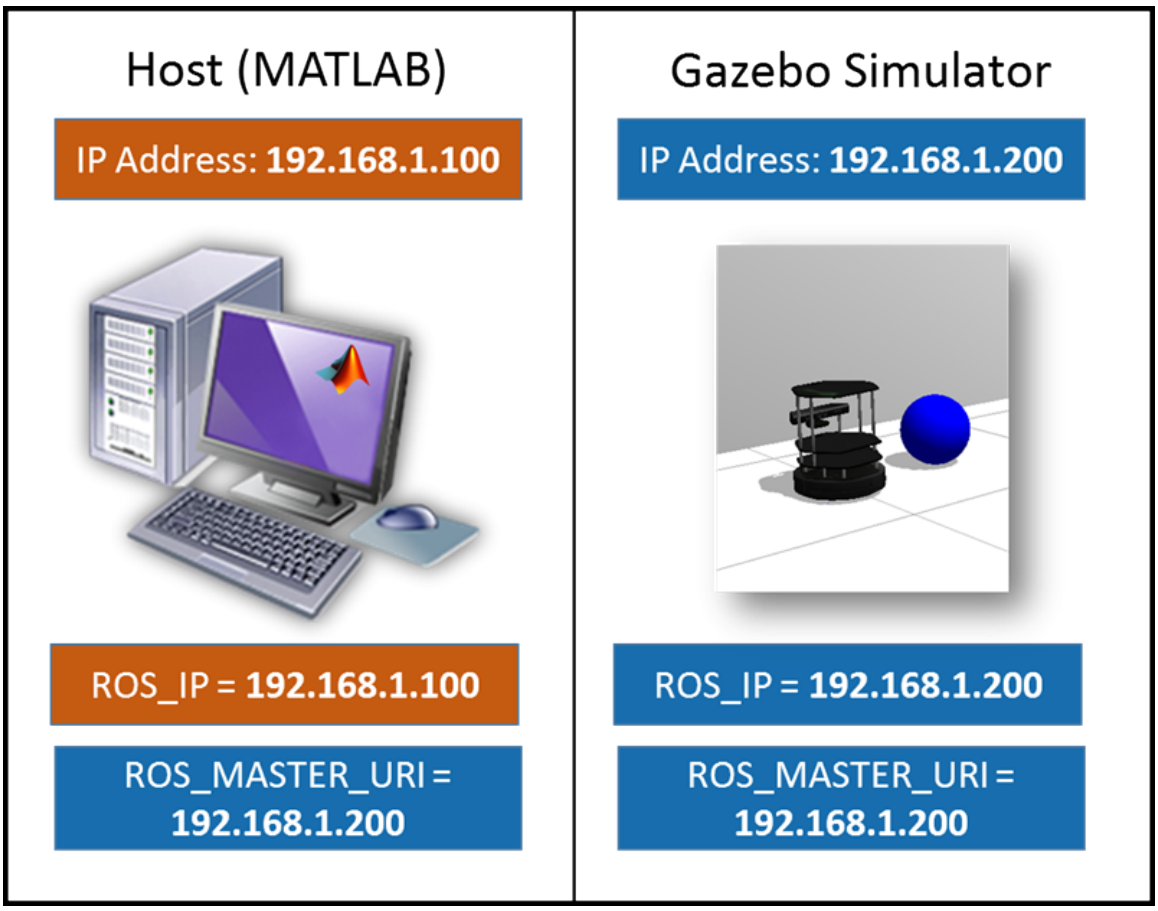


Figure 5.8: IP message communication flow between MATLAB running on a physical machine and Gazebo running on a virtual machine. ROS handles the communication by connecting these applications through a Uniform Resource Identifier (URI) to label the applications with their IP addresses. As is shown, the ROS_MASTER_URI runs on both machines and handles the data flow.

CHAPTER 6: RESULTS

6.1 Similar Study Results

In order to discuss the results of the procedure described here it is beneficial to look at similar results in the field. By "similar" it is meant that these studies used the same dataset used in this research and attempted to perform the same function as this research, classification of events within said dataset, with varying methods. The metric for comparison, accuracy, is defined as the rate of correct classification of an event; false positives are also counted as incorrect classification for this metric. While no other paper details this particular technique, there have been many attempts to reach a similar outcome. A short list of these results are identified from [47] and are compiled in Table 6.1 with the best performance of Subject 25.

Table 6.1: Comparison of Similar Methods

Comparable Methods	
Method	**Accuracy**
Almoari [48]	74.97%
Sun [49]	65.00%
Shenoy [50]	82.06%
Tolic [51]	68.21%
Previous work	72.56%
Current Method	96.36%

Similar methods to the one proposed here obtain a 72.56% correct classification accuracy [52]; false classifications are also factored into these performance reports.

As it is not mentioned how many subjects each method was applied to in order obtain the results in Table 6.1, the proposed method of this dissertation applied the method to 10 randomly chosen subjects of the 109 subjects present in the dataset. As a representative of these 10 subjects, one subject will be presented for the results, Subject 25. The performance of this representative is typical of the results of the other subjects and highlights trends seen in the results.

6.2 Analyzing All 64 Electrodes

The first experiment that shall be discussed is the effect of starting with all 64 electrodes of the cap as data and the effects of the method upon reducing the dimension of the entire dataset as it relates to performance and training time.

Table 6.2 shows the results of using the proposed method with different parameters, but all starting with 64 electrodes. The parameters that were changed through the trials were the PCA dimension, ICA dimension, hidden layer count of the neural network, overall performance, and how many epochs it took to train the neural network. Conventionally, the training time is discussed in epochs. An epoch is defined as the time frame a certain signal is present. This is useful in the field of EEG and BCI classification because an epoch gives a constant window that is known. In the dataset each epoch is defined as being 4 seconds long at 160 Hz, meaning each epoch contains $4(s)*160(Hz) = 640$ data points. Increasing the hidden layer count, the intermediate steps shown in Figure 2.6, beyond what is presented here has diminishing returns, with the typical optimal number of hidden layers being between 15 and 20. Since the results in Table 6.1 that use neural networks all keep the number of hidden layers small the trials in Table 6.2 are grouped in to sets of 3, with an increasing number of hidden layers without changing the PCA or ICA dimensionality; the only exception to this is trial 10, where 20 hidden layers is shown to highlight the diminishing returns of increasing the layer count and to also show an interesting trend, which will be discussed.

Examining trials 1, 2, and 3 as a baseline the performance does not change significantly with a change of hidden layers, though it is important to note that the number of epochs it takes to train the neural network increases as the layer count increases.

Trials 4, 5, and 6 perform only an ICA on the data. This does not impact the performance for correct classification in a meaningful way, but it had the benefit of greatly reducing the number of epochs it takes to train the network. The trend for this training is: The more hidden layers in the network, the more epochs it takes to train the network.

Trials 7, 8, 9, and 10 introduce the full proposed method by performing PCA before the ICA stage. Comparing trial 7 with 1 and 4, trial 8 with 2 and 5, and trial 9 with 3 and 6 there is a consistent increase in performance. By trial 9, there is an overall increase of performance of 6.02%, on average. An interesting note is that adding in the PCA inverses the trend of number of epochs needed when the PCA does not reduce the dimensionality. Observing trial 10 it can be seen that there is not a large increase in performance by increasing the number of hidden layers, although it does reduce the number of epochs needed to train the network.

Table 6.2: Sample of Results with Processing 64 Starting Electrodes

Subject 25: 64 Starting Electrodes					
Trial	**PCA Dimension**	**ICA Dimension**	**Hidden Layer Count**	**Performance (%)**	**Epoch**
1	N/A	N/A	5	90.16	311
2	N/A	N/A	10	90.85	326
3	N/A	N/A	15	89.96	423
4	N/A	64	5	88.10	193
5	N/A	64	10	89.20	243
6	N/A	64	15	90.17	328
7	64	64	5	91.20	236
8	64	64	10	94.34	123
9	64	64	15	96.08	119
10	64	64	20	96.36	105
11	32	32	5	84.58	171
12	32	32	10	89.99	316
13	32	32	15	90.78	323
14	16	16	5	75.44	132
15	16	16	10	77.84	167
16	16	16	15	78.92	203

Starting with trial 11 the PCA is used to try and reduce the dimension of the data; in doing so the overall performance suffers when compared to the PCA and ICA of the original dimensions, but once the hidden layers are increased to 15 the performance is about the same as no ICA or ICA alone. The benefit of the approach is a generally faster training time. Reducing the dimension even further continues this trend. The

comparison of reducing the dimension to 16 elements from 64 is compared to starting with only 16 dimensions in section 6.5.

6.3 Analyzing 16 Electrodes

Table 6.3 contains the results from Subject 25 when starting with the 16 electrodes outlined in [52] and highlighted in blue in Figure 6.1. The reason that only these electrodes need to be observed is due to their placement directly above the motor cortex of the brain, where muscular movement is generated.

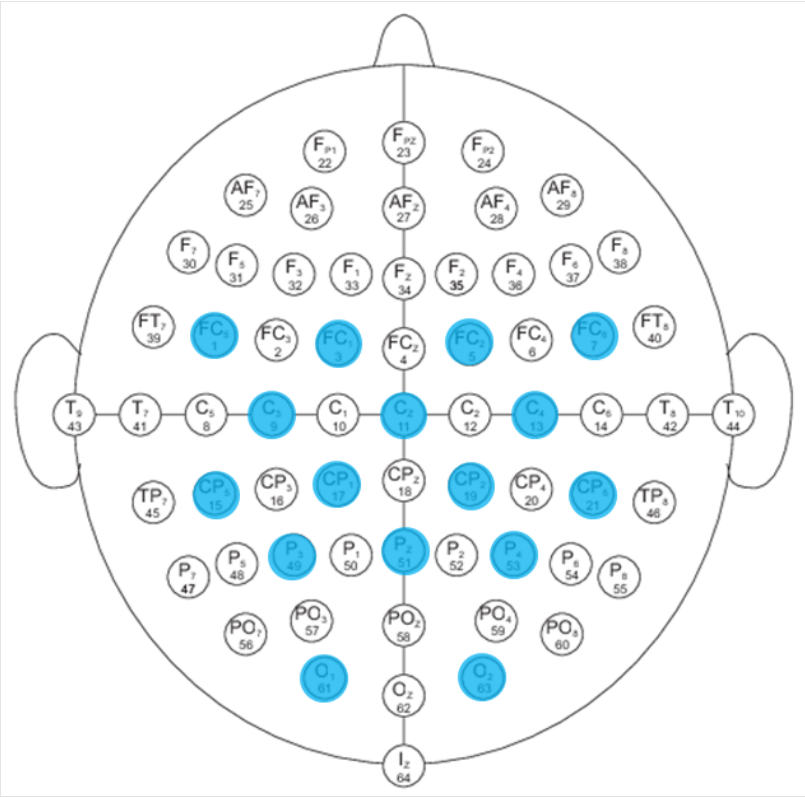


Figure 6.1: 16 Electrodes over the motor cortex used to detect movement intent for study.

Trials 1, 2, and 3 observe the trend of increasing the layer count for the neural network on the signals. Predictably, the performance and number of training epochs increases as the layer count increases. It should be noted that the performance increase at this stage is noticeably greater than the same trials in Table 6.2.

Introducing the ICA into the system produces a slight increase in performance

at 1.45%. At this stage the network is much quicker to train at lower hidden layer counts, though this does not last as the count increases.

Table 6.3: Sample of Results with Processing 16 Starting Electrodes

Subject 25: 16 Starting Electrodes					
Trial	**PCA Dimension**	**ICA Dimension**	**Hidden Layer Count**	**Performance (%)**	**Epoch**
1	N/A	N/A	5	77.70	295
2	N/A	N/A	10	79.62	276
3	N/A	N/A	15	83.26	374
4	N/A	16	5	77.09	130
5	N/A	16	10	81.29	283
6	N/A	16	15	84.80	382
7	16	16	5	77.26	330
8	16	16	10	80.91	195
9	16	16	15	86.80	176
10	8	8	5	71.09	146
11	8	8	10	73.32	169
12	8	8	15	75.02	244

With the whole method applying in trials 7, 8, and 9 the system reaches its peak performance values; with an overall increase in performance by 3.54% over no processing and 2.00% over just ICA processing. As was the case when starting with 64 electrodes, increasing the number of hidden layers decreases the training time for the neural network.

Reducing the dimensionality in trials 10, 11, and 12 reduces the performance of the system overall, but eventually approaches the performance of trials 1, 4, and 7.

6.4 Baseline Neural Network Performance

This section will present a comparative analysis of Table 6.5, Table 6.2, and Table 6.3 to more closely show the relationship in the trends that emerge in the processing of the signals.

Table 6.4 compares the training of the neural network with no PCA or ICA processing. These results are a baseline that was used to evaluate the performance with other methods. As can be seen in the table there is an overall greater performance when starting with 64 electrodes, as opposed to 16 electrodes, most likely due to the greater resolution that more data provides. Also, while there was no significant increase in performance when starting with 64 electrodes, additional hidden layers provided a 5.56% increase when only 16 electrodes worth of data was present.

Table 6.4: Comparing Neural Network Performance with no Technique

Subject 25: Comparison No Techniques					
Starting Electrodes	**PCA Dimension**	**ICA Dimension**	**Hidden Layer Count**	**Performance (%)**	**Epoch**
64	N/A	N/A	5	90.16	311
16	N/A	N/A	5	77.70	295
64	N/A	N/A	10	90.85	326
16	N/A	N/A	10	79.62	276
64	N/A	N/A	15	89.96	283
16	N/A	N/A	15	83.26	374

6.5 Analysis of Research Results

Table 6.5 groups the results of the method when PCA and ICA were used, but the dimension was not reduced beyond the starting number of electrodes. Using this

method there was an increase of performance of 5.16% using 64 electrodes and 9.63% using 16 electrodes. This table is of special note, as when there is no dimensionality reduction of the data, the number of epochs needed to train the increase in performance decreased with the addition of hidden layers. There was a limit to this, as can be seen in the final row of the table where the epochs increased, as was the case of other methods.

Table 6.5: Comparing Results with No Dimensionality Reduction on Different Number of Starting Electrodes

Subject 25: Comparison No Reduction					
Starting Electrodes	**PCA Dimension**	**ICA Dimension**	**Hidden Layer Count**	**Performance (%)**	**Epoch**
64	64	64	5	91.20	236
16	16	16	5	77.26	330
64	64	64	10	94.34	123
16	16	16	10	80.91	195
64	64	64	15	96.08	119
16	16	16	15	86.80	176
64	64	64	20	96.36	105
16	16	16	20	86.89	203

Finally, Table 6.6 shows how the performance when reducing the dimensionality to 16 from 654 electrodes compares to starting with 16 electrodes. With only 5 hidden layers in the network the performance is relatively close, 1.82% apart, but the technique that started with 64 electrodes took less than half of the time to train than starting with the base dimensions.

Moving to 10 hidden layers the performance gap widened to 3.07% with only a

difference of 28 epochs in training time. This is due to the trend of adding hidden layers to reduced dimension data needing more time to train and unreduced dimension data needing less time to train.

The last two rows show the widening gap even more, with the number of epochs to train the network for the dimension reducing technique overtaking the base 16 electrodes. With this comparison of performance values it can be said that reducing the 64 electrodes to fewer dimensions starts with similar results, but is not as robust as starting with that number of dimensions.

Table 6.6: Comparing Dimensionality Reduction with Similar Starting Dimensions

Subject 25: 16 Electrodes from Reduction VS No Reduction					
Starting Electrodes	**PCA Dimension**	**ICA Dimension**	**Hidden Layer Count**	**Performance (%)**	**Epoch**
64	16	16	5	75.44	132
16	16	16	5	77.26	330
64	16	16	10	77.84	167
16	16	16	10	80.91	195
64	16	16	15	78.92	203
16	16	16	15	86.80	176

6.6 Generalized Neural Network

As an extension of this work there was an attempt to build a more generalized neural network for many subjects. The intent of this procedure was to use ten subjects as the training set and evaluate performance, individually, on ten different subjects to observe how a BCI trained on data that is not part of the individual will perform.

In developing a BCI the standard approach is to train an individual, much like in physical therapy, to perform the action in order to develop the neural pathways for

Table 6.7: Generalizing the BCI Neural Network

Generalized Neural Network	
Trial	**Performance (%)**
Network	98.26
Highest	78.01
Lowest	55.96
Average	69.98

the independent BCI. Moving forward in the field of BCI development it is desirable to have a generalized model that has acceptable base performance as opposed to starting from the beginning with each subject. Much like the early development of speech recognition there will be a shift in the future to a model that doesn't need to be trained on a per-person basis.

Table 6.7 gives an overview of how the BCI performs with the ten new subjects. The training data used was the ten previous subjects in the study, with ten new subjects randomly chosen for evaluation. There is no overlap in the training data with the evaluation data. "Network" refers to the performance of the trained neural network data validation set. Since there is ten times the amount of training data than in previous chapters and each misclassification is weighted less, the performance is overall better than in previous chapters.

The neural network was trained using the best approach found in the results of this research. The inputs were passed through a lowpass filter, evaluated with a PCA of 64 dimensions, ICA of 64 dimensions, and the network utilized 20 hidden layers.

The highest performance in the resulting data was from Subject 43, with a 78.01% correct classification rate, and the lowest performance was Subject 19 with 55.96%. Overall, the average performance was 69.98%. Comparing these results with Table 6.1, the generalized model performs about as well as similar specialized methods.

CHAPTER 7: CONCLUSION

The novel technique presented in this dissertation was an expansion upon previous research and developed a useful tool for the development of brain computer interfaces. This technique was compared with current state of the art techniques for intent classification and was shown to be superior in accurate classification of imagined hand grasping. The application of the methodology of using PCA then ICA on EEG signals increases the overall performance of a BCI and in the case where dimensionality is not reduced in the PCA stage reduces the training time of the system.

Most of the goals for this research were met, including:

- Identification of modern BCI techniques and architectures:

  As covered in this work, a BCI consists of a set of hardware and software that interprets electromagnetic impulses from the brain to actuate a physical device. This research focused on the software portion of that connection and as such presented different techniques currently used in the field to train BCIs.

- Interfacing MATLAB, EEGLAB for processing signals:

  A set of functions was created to allow the interaction of MATLAB, EEGLAB, and the neural network toolbox within MATLAB. This included the pre-filtering of the signals before they were loaded into EEGLAB for graphing and epoch framing, the stage where the events were linked with the time stamps included in the data. Exporting from EEGLAB, PCA and ICA parameter analysis was processed on the data. Finally the sweep of neural network hidden nodes was automated.

- Saving of results for comparative analysis:

Once the training of the network was complete the results were saved to excel files for later comparison. Various graphs, such as performance and region of convergence were saved from the output of each hidden layer analysis.

7.1 Future Work

The only goal of the research that was not met was the simulation of a robotic hand in a ROS environment. As the researched was focused on the development and performance analysis of the software technique time did not permit simulation being added. This step would be beneficial as a more intuitive analysis to the end researcher on the performance of the device. The current algorithm would be used as shown in Figure 7.1. The algorithm would be loaded in MATLAB after the training stages have been completed. Online, real-time, signals would then be streamed into MATLAB via the use of an EEG cap with electrodes. The system would then parse the input in to 4 second epochs at 160 Hz. These epochs would be processed by the algorithm which would then send a signal to the ROS core, identifying the classification of the signal as a left, right, or no grasp. The ROS core would publish to the corresponding node to control the output, that being either a simulator or a prosthetic hand.

Additional benefit could be provided through the use of a graphical user interface to abstract the functions created as a part of this research. This would enable the code to be shared with a greater number of researchers and would help in the dissemination of the technique.

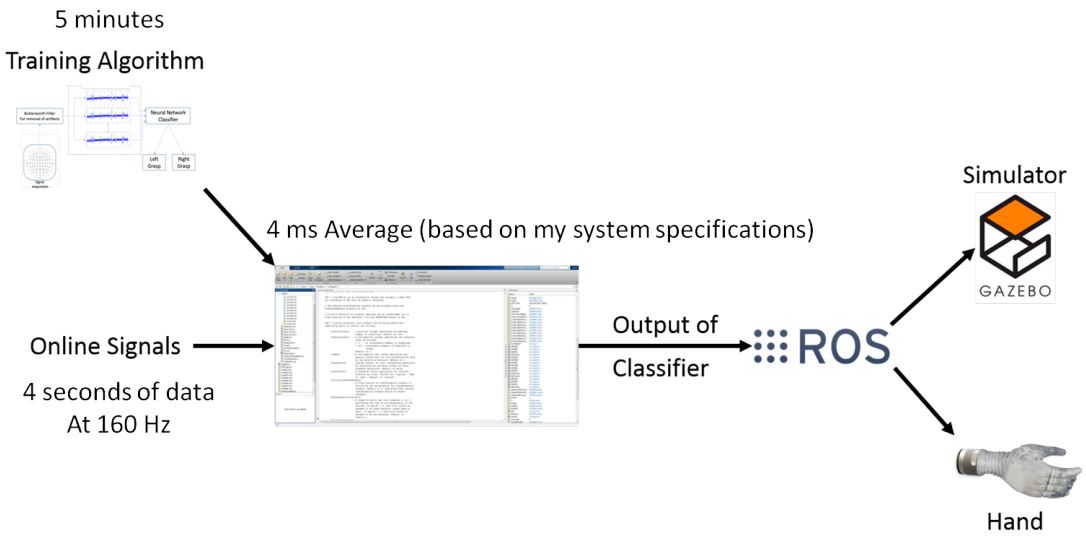


Figure 7.1: System flow of final design for deployment. The training algorithm is fed in to MATLAB with the online, real-time, signals as the input. The algorithm classifies the signal as either a left or right hand grasp and sends a signal to the ROS core [8]. This is then either sent to the simulator [9] or a prosthetic limb [10].

REFERENCES

[1] G. Mislap, "P300 Speller Example." Available at: https://www.youtube.com/watch?v=08GNE6OdNcs.

[2] B. Blaus, "Medical gallery of Blausen Medical 2014," *WikiJournal of Medicine*, vol. 1, no. 2, 2014.

[3] U. Herwig, P. Satrapi, and C. Schönfeldt-Lecuona, "Using the International 10-20 EEG System for Positioning of Transcranial Magnetic Stimulation.," *Brain topography*, vol. 16, pp. 95–9, Jan 2003. Available at: http://www.ncbi.nlm.nih.gov/pubmed/14977202.

[4] Z. Weinberg, "File:Svm separating hyperplanes (SVG).svg - Wikimedia Commons," 2012.

[5] P. Buch, "File:Svm max sep hyperplane with margin.png - Wikimedia Commons," 2011.

[6] V. Jurcak, D. Tsuzuki, and I. Dan, "10/20, 10/10, and 10/5 Systems Revisited: Their Validity As Relative Head-Surface-Based Positioning Systems.," *NeuroImage*, vol. 34, pp. 1600–11, Feb 2007.

[7] DARPA, "Hand Proprioception & Touch Interfaces." Available at: http://gazebosim.org/haptix.

[8] ROS, "Open source robotics foundation." Available at: https://www.ros.org.

[9] Gazebo, "Open source robotics foundation." Available at: https://www.gazebosim.org.

[10] Mobius Bionics LLC, "All about the Luke Arm." Available at: http://www.mobiusbionics.com/the-luke-arm.html.

[11] H. BEKCER, "Uber das Elektrenkephalogramm des Menschen. IV," *Mitteilung. Arch. f. Psychiat*, vol. 278, no. 1875, 1932.

[12] G. Townsend, B. K. LaPallo, C. B. Boulay, D. J. Krusienski, G. E. Frye, C. K. Hauser, N. E. Schwartz, T. M. Vaughan, J. R. Wolpaw, and E. W. Sellers, "A Novel P300-Based Brain-Computer Interface Stimulus Presentation Paradigm: Moving Beyond Rows and Columns.," *Clinical Neurophysiology*, vol. 121, pp. 1109–20, Jul 2010.

[13] E. W. Sellers, T. M. Vaughan, and J. R. Wolpaw, "A Brain-Computer Interface for Long-Term Independent Home Use," *Amyotrophic Lateral Sclerosis*, vol. 11, pp. 449–55, Oct 2010.

[14] M. Velliste, S. Perel, and M. Spalding, "Cortical Control of a Prosthetic Arm for Self-Feeding," *Nature*, vol. 453, pp. 1098–101, Jun 2008.

[15] D. Taylor, S. Tillery, and A. Schwartz, "Direct Cortical Control of 3D Neuroprosthetic Devices," *Science*, vol. 296, pp. 1829–1832, Jun 2002.

[16] A. Schwartz, X. Cui, D. Weber, and D. Moran, "Brain-Controlled Interfaces: Movement Restoration with Neural Prosthetics," *Neuron*, vol. 52, pp. 205–20, Oct 2006.

[17] J. L. Collinger, B. Wodlinger, J. E. Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber, A. J. C. McMorland, M. Velliste, M. L. Boninger, and A. B. Schwartz, "High-Performance Neuroprosthetic Control by an Individual with Tetraplegia.," *Lancet*, vol. 381, pp. 557–64, Feb 2013.

[18] A. Kübler, B. Kotchoubey, and J. Kaiser, "Brain Computer Communication: Unlocking the Locked In.," *Psychological*, 2001.

[19] J. Conrad, S. Kuyath, and T. Major, "Developing a Brain Computer Interface Control System for Robotic Movement in Two Dimensions," 2014.

[20] T. Mulholland, "Human EEG, Behavioral Stillness and Biofeedback.," *International journal of psychophysiology : official journal of the International Organization of Psychophysiology*, vol. 19, pp. 263–79, May 1995.

[21] M. Linden, T. Habib, and V. Radojevic, "A Controlled Study of the Effects of EEG Biofeedback on Cognition and Behavior of Children with Attention Deficit Disorder and Learning Disabilities.," *Biofeedback and self-regulation*, vol. 21, pp. 35–49, Mar 1996.

[22] T. O. Zander and C. Kothe, "Towards Passive Brain-Computer Interfaces: Applying Brain-Computer Interface Technology to Human-Machine Systems in General.," *Journal of neural engineering*, vol. 8, p. 025005, Apr 2011.

[23] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, "Quadcopter Control in Three-Dimensional Space Using a Noninvasive Motor Imagery-Based Brain-Computer Interface," *Journal of Neural Engineering*, vol. 10, p. 46003, Aug 2013.

[24] Y. Wang, X. Gao, B. Hong, C. Jia, and S. Gao, "Brain-Computer Interfaces Based on Visual Evoked Potentials.," *IEEE engineering in medicine and biology magazine*, vol. 27, no. 5, pp. 64–71, 2008.

[25] L. Farwell and E. Donchin, "Talking Off the Top of Your Head: Toward a Mental Prosthesis Utilizing Event-Related Brain Potentials.," *Electroencephalography and clinical neurophysiology*, vol. 70, pp. 510–23, Dec 1988.

[26] E. Donchin, "The Mental Prosthesis: Assessing the Speed of a P300-Based Brain-Computer Interface," *IEEE Transactions on*, vol. 8, no. 2, pp. 174–179, 2000.

[27] E. C. Leuthardt, K. J. Miller, G. Schalk, R. P. N. Rao, and J. G. Ojemann, "Electrocorticography-Based Brain Computer Interface–the Seattle Experience.," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 14, pp. 194–8, Jun 2006.

[28] N. K. Logothetis, "The Neural Basis of the Blood-Oxygen-Level-Dependent Functional Magnetic Resonance Imaging Signal.," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 357, pp. 1003–37, Aug 2002.

[29] N. Weiskopf, K. Mathiak, S. W. Bock, F. Scharnowski, R. Veit, W. Grodd, R. Goebel, and N. Birbaumer, "Principles of a Brain-Computer Interface (BCI) Based on Real-Time Functional Magnetic Resonance Imaging (fMRI).," *IEEE transactions on bio-medical engineering*, vol. 51, pp. 966–70, Jun 2004. Available at: `http://www.ncbi.nlm.nih.gov/pubmed/15188865`.

[30] S. Yoo, T. Fairneny, N. Chen, and S.-E. Choo, "Brain-Computer Interface Using fMRI: Spatial Navigation by Thoughts," *Neuroreport*, vol. 15, pp. 1591–1595, Jul 2004.

[31] D. G. Nair, K. L. Purcott, A. Fuchs, F. Steinberg, and J. a. S. Kelso, "Cortical and Cerebellar Activity of the Human Brain During Imagined and Executed Unimanual and Bimanual Action Sequences: a Functional MRI Study.," vol. 15, pp. 250–60, Feb 2003.

[32] R. C. Oldfield, "The Assessment and Analysis of Handedness: the Edinburgh Inventory.," *Neuropsychologia*, vol. 9, pp. 97–113, Mar 1971. Available at: `http://www.ncbi.nlm.nih.gov/pubmed/5146491`.

[33] R. Sitaram, H. Zhang, C. Guan, M. Thulasidas, Y. Hoshi, A. Ishikawa, K. Shimizu, and N. Birbaumer, "Temporal Classification of Multichannel Near-Infrared Spectroscopy Signals of Motor Imagery for Developing a Brain-Computer Interface.," *NeuroImage*, vol. 34, pp. 1416–27, Feb 2007.

[34] J. Mellinger, G. Schalk, C. Braun, H. Preissl, W. Rosenstiel, N. Birbaumer, and A. Kübler, "An MEG-Based Brain-Computer Interface (BCI).," *NeuroImage*, vol. 36, pp. 581–93, Jul 2007.

[35] T. Mladenov, K. Kim, and S. Nooshabadi, "Accurate Motor Imagery Based Dry Electrode Brain-Computer Interface System for Consumer Applications," *2012 IEEE 16th International Symposium on Consumer Electronics*, pp. 1–4, Jun 2012.

[36] S. Makeig and J. Onton, "ERP Features and EEG Dynamics: an ICA Perspective," *Oxford Handbook of Event-Related Potential*, pp. 1–58, 2009.

[37] A. Delorme, J. Palmer, J. Onton, R. Oostenveld, and S. Makeig, "Independent EEG Sources are Dipolar.," *PloS one*, vol. 7, p. e30135, Jan 2012.

[38] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[39] A. Hyvärinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[40] A. Goldberger, L. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet : Components of a New Research Resource for Complex Physiologic Signals," *Circulation*, vol. 101, pp. e215–e220, Jun 2000.

[41] G. Schalk, D. J. Mcfarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000 : A General-Purpose Brain-Computer Interface (BCI) System," vol. 51, no. 6, pp. 1034–1043, 2004.

[42] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," *Conference Proceedings - IEEE Southeastcon*, pp. 243–249, 2011.

[43] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, "Premotor Cortex and the Recognition of Motor Actions," vol. 3, pp. 131–141, 1996.

[44] DARPA, "Hand Proprioception & Touch Interfaces (HAPTIX) - DARPA-BAA-14-30 (Archived) - Federal Business Opportunities: Opportunities." Available at: `https://www.fbo.gov/index?s=opportunity&mode=form&id=ccbc51af982e354304886e363e6c8404&tab=core&_cview=1`.

[45] G. Langevin, "InMoov | open-source 3D printed life-size robot." Available at: `http://inmoov.fr/`.

[46] MATHWORKS, "Get Started with Gazebo and a Simulated TurtleBot - MATLAB & Simulink Example." Available at: `https://www.mathworks.com/help/robotics/examples/get-started-with-gazebo-and-a-simulated-turtlebot.html`.

[47] U. Sydney, L. Yao, and Q. Z. Sheng, "Neural Information Processing," vol. 10638, no. Oct, 2017.

[48] M. H. Alomari, A. Abubaker, A. Turani, A. M. Baniyounes, and A. Manasreh, "EEG Mouse : A Machine Learning-Based Brain Computer Interface," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 5, no. 4, pp. 193–198, 2014.

[49] L. Sun and Z. R. Feng, "Classification of Imagery Motor EEG Data with Wavelet Denoising and Features Selection," *International Conference on Wavelet Analysis and Pattern Recognition*, vol. 2016-November, pp. 184–188, 2016.

[50] H. V. Shenoy, A. P. Vinod, and C. Guan, "Shrinkage estimator based regularization for EEG motor imagery classification," in *2015 10th International Conference on Information, Communications and Signal Processing, ICICS 2015*, 2016.

[51] M. Tolić and F. Jović, "Classification of Wavelet Transformed Eeg Signals With Neural Network for Imagined Mental and Motor Tasks," *Kinesiology: Classification of Wavelet Transformed EEG*, vol. 45, no. 1, pp. 130–138, 2013.

[52] T. C. Major and J. M. Conrad, "The Effects of Pre-Filtering and Individualizing Components for Electroencephalography Neural Network Classification," *Conference Proceedings - IEEE Southeastcon*, 2017.