University of Strathclyde

Department of Bioengineering

MSc Thesis Project

Extracting Signatures from Muscle Activity Data for Control of Myoelectric Prosthesis

Author:
Derek Elofson

Supervisor:
Dr. Heba Lakany

September 4, 2012

Abstract

Upper limb myoelectric prostheses are controlled by the voluntary contraction of residual muscles of the amputated limb. These contractions produce weak electric signals that can be captured using surface electromyography (sEMG) which can be used to control the prosthesis. A number of multi-articulate prostheses have recently been developed that utilise sEMG control, offering the potential for increased usability. The intent is that the patient will be able to reproduce the same muscle contractions that they would have normally without the prosthesis in order for it to perform the associated hand gesture or finger movement [7].

In this project, we explored and investigated the existence of sEMG signatures associated with particular hand gestures and finger movements. Experiments were designed to record sEMG activity from healthy subjects. Our experiments used four sensors instead of the two that the current model uses, due to limitations in pattern recognition with only two sensors. We looked into signal processing and pattern recognition methods to find if there is any commonality amongst the different subjects so that the iLIMB could identify the gestures that someone makes without having to program it for each individual [14]. The features we found showed promise for future developments in pattern recognition of myoelectric signals.

The multi-articulate prosthesis we used was the Touch Bionics iLIMB. This is an advanced myoelectric prosthesis where each digit is individually powered, allowing the prosthesis to be positioned into several gestures. The current model uses two electrodes, one placed on the extensor muscle and the other on the flexor muscle for EMG control.

Contents

1 Introduction

1.1 Aims

Prostheses can give people that have lost a limb some of their previous mobility and functions back to them. Our hands are often the first thing we use to interact with our environment. So many everyday actions require a normal functioning hand, from eating to typing on a keyboard. Because of this, loss of the forearm and hand can have massive effects on how a person interacts with their surroundings. Due to the complexity of the hand and its wide range of motions and gestures, providing an adequate replacement when someone loses theirs has proven to be a unique challenge.

Myoelectric prostheses that have individually powered fingers have recently been developed by a number of companies. Since each finger can be individually positioned irrespective of the others, these prostheses have the potential to restore a range of motions and abilities to the subject that previous prostheses could never accomplish. Myoelectric prostheses utilise surface electromyography (sEMG) as the way to control the prosthesis. Most current models have the patient perform a certain series of muscle contractions in order to signal for a certain hand gesture. Often times however, these series of muscle contractions may be unintuitive or unrelated to the actual desired gesture. The aim of this study is to improve on the pattern recognition capabilities of myoelectric prostheses in order to make it easier and more intuitive for the subject to perform the desired gestures. [7]

The future of myoelectric prostheses involves more research into EMG signal pattern recognition and its application to control of the prosthesis. Other avenues of research involve utilising sensory feedback, which would greatly increase the acceptance of myoelectric prostheses and further restore the subjects capabilities [26]. However, this is outside the scope of this study and should be looked at some other time.

1.2 Thesis Hypothesis

We hypothesise that surface electromyography signals are capable of being used as a control signal in a multi-articulate forearm prosthesis so that the subject only has to contract their muscles in a natural manner for performing the desired gesture. In addition, we will show that use of four sensors will bring benefits to gesture discrimination that would improve pattern recognition techniques for control of these prostheses.

2 Upper Limb Prostheses

2.1 Introduction

There are several potential causes that would require a person to need a prosthesis, but for the majority of people this would be either congenital or some form of trauma that required an amputation of the arm. Regardless of the reason, the persons ability to interact with their surroundings would be much more limited than that of a healthy person.

2.1.1 Necessities for Prostheses

There are several reasons why someone would have to amputate a limb. Infection and circulatory problems account for a large amount of amputations, however this is much more prevalent in the lower limbs. [9] Trauma is much more likely a cause of forearm amputation, along with other diseases that can cause problems with normal limb functions.

People with congenital limb deficiencies also often require the use of a prosthesis. Currently it is estimated that there will be about sixty people born in the UK each year that will have some sort of congenital limb deficiency that will require them to use a prosthesis.[8] Causes for congenital deformities like this are not very well understood. Genetics is one possible cause, but it is probably not the only factor. Environment and possible chemicals that are present can also contribute to some of the defects, but unfortunately not enough data is present to determine most causes.

Congenital limb deficiencies can present as missing entire limbs right from birth. However, it is more common for someone to be born with a partially formed limb with the hand and part of the form arm not present. This proves to be much more beneficial for controlling a prosthesis, and essential for controlling a myoelectric prosthesis without requiring drastic surgery.

2.1.2 Current Models of Upper Limb Prostheses

The more traditional forearm prostheses are much simpler than the current state-of-the-art powered and myoelectric prostheses. One that is commonly seen utilises a hook at the end where the hand would be. This hook is attached to a cable, whose other end is connected to a harness the subject is wearing. When the subject extends their arm forwards, this puts a tension on the cable and causes the hook to open. Lowering the arm again relieves the tension and allows the hook to close. This mechanism is designed to allow people some ability to grasp objects, but doesnt afford much flexibility or capabilities other than this.

There are also prostheses that are mainly cosmetic in nature. These attempt to provide the semblance of a normal hand where the subject is missing theirs. These often do not have any functionality and are only made to look realistic.

3 Surface Electromyography

3.1 Introduction

Electromyography (EMG) is the process of reading the electrical signals sent to a skeletal muscle to initiate a contraction. EMG can be used for many purposes, ranging from diagnosing abnormal muscle activity to determining muscle characteristics. This is somewhat similar to electrocardiography (ECG) or electroencephalography (EEG), where we want to monitor certain activity. For our purposes, we are looking at the uses of EMG signals for control of powered prostheses. Firstly, it would be useful to understand how myoelectric signals are formed, where they originate, and any other characteristics we can glean from them that would help in utilising them as control signals.

3.2 Physiology of Myoelectric Signals

Skeletal muscles are generally muscles that are controlled by activity originating from the central nervous system (CNS). Nervous tissue in the CNS generate what is known as an action potential (AP) to signal other nerves. Changing chemical gradients of sodium and potassium across the nerve membrane cause a propagation of voltage depolarisation along a nerves axon, which then quickly repolarises. This depolarisation impulse travels along the axon towards the end, called the synapse. If the synapse ends at a neuromuscular junction, the synapse will release acetylcholine (ACh) into the muscle sarcolemma, the cell membrane of the muscle tissue. This neurotransmitter binds to receptors on the sarcolemma which will then generate a new AP. This AP will propagate throughout the muscle cell and cause it to contract at all points simultaneously. These action potentials are essentially what we are recording with EMG. [21, 20]

A nerve cell, like all living cells, has a transmembrane potential. When speaking of potentials, we are referring to the electrical potential caused by a chemical gradient across the cell membrane. The potential across the membrane when the cell is at rest is referred to as the resting potential. In most neurons the resting potential is about -70mV. This potential is created by a relatively high concentration of positively charged potassium ions (K^+) in the intracellular fluid of the cell. Similarly, there are positively charged sodium ions (Na^+) in the extracellular fluid. However, this concentration is less than the K^+ ions and so sets up an ionic difference across the membrane. The intracellular fluid, due to its higher concentration of K^+ ions, will have a greater positive ionic charge than the extracellular fluid. This is what causes the potential difference. [20]

Due to the chemical gradients across the membrane, both ions have a tendency to move to the opposite side of the membrane through leakage channels. Sodium-potassium exchange pumps that use up energy work to maintain the chemical gradients in the resting state. However, a stimulus from another neurons synapse will change the membrane potential. This is then called a graded potential. Neurotransmitters from the synapse will activate extra channels in the membrane, allowing the inflow of Na^+ ions into the cell. This will alter the membrane potential in the vicinity of the synapse connection, but will taper off the

further from the site of stimulation. This inflow of Na^+ ions will cause a depolarisation of the membrane potential, moving it closer to 0mV. [20] If the graded potential is close enough to the axon hillock, this may generate an action potential. For this to happen, the graded potential must make the axon hillock reach a threshold potential of about -60mV. When this happens nearby Na^+ gates that are activated only when the voltage reaches this point or higher will open, allowing more Na^+ ions to flow in and further depolarise the potential. The transmembrane potential will eventually reach a positive value of about +30mV. At this point the Na^+ channels will close again and K^+ channels will open, allowing K^+ ions to flow out of the membrane due to the electrical and chemical gradient. This process repolarises the membrane until it reaches about resting potential. [20]

The AP is propagated along the axon by this means of depolarising the adjacent membrane ahead of it while the membrane behind it works to repolarise. These propagate along the axon and terminate at a synapse. The AP then causes the synapse to release a neurotransmitter into the extracellular fluid. If the synapse is stimulating a skeletal muscle, it will be releasing acetylcholine (ACh), which will bind to receptors on the muscle membrane. This in turn will generate another AP, but this time it will propagate throughout the muscle cell which will cause the cell to contract. These APs travelling along the muscle cell are what we are monitoring with EMG readings. It is good to note that the resting potential of a muscle cell is closer to -90mV. Knowing this can tell us where our baseline for non-activity should be.

Seeing that the voltages involved in AP generation and muscle contraction are quite low, we need to understand that we need a relatively sensitive sensor in order to pick up these signals. It is also important to note that there is a large possibility for noise to interfere in the sensor, so we need to take care in filtering out any unwanted signals.

3.3 Methods of Recording

There are currently two ways of obtaining the signals for EMG. Intramuscular EMG involves the insertion of a fine needle directly into the body of the muscle. This technique is usually used when diagnosing abnormal muscle activity or characterising a muscle in a laboratory setting. Intramuscular EMG would not be suitable for use in a prosthesis considering that we want the subject to be more physically capable than without the prosthesis. Having a subject be physically active while they have several needles inserted into their muscles would likely prove very uncomfortable. In addition, the intramuscular needle would only be penetrating one muscle fibre at a time, when it would be more beneficial to us to see the activity of an entire motor unit to help determine overall contraction intensity.

The other technique available is surface electromyography (sEMG), which is what modern myoelectric prostheses utilise to monitor muscle activity. This involves the use of electrodes that are placed on the skin surface above the muscle to be monitored. This method is non-invasive, and so should be much more comfortable for the subject and more portable for use in a prosthesis. For use in a myoelectric prosthesis, we want to not only be able to monitor when a certain muscle contracts but also the intensity of contraction. sEMG should give us a more generalised perspective on the entire motor unit rather than just individual muscle fibres which allows us to determine contraction intensity.

The sensor we will be using is a proprietary one manufactured by Touch Bionics for use in their iLIMB prosthesis. The sensor is a reusable bipolar sEMG device that has three contacts on it that sit on the surface of the skin. This particular sensor utilises a differential amplifier in order to obtain and increase the myoelectric signal. Two of the contacts on the sensor surface go to the differential amplifier, while the third contact is used as a voltage reference. This particular configuration of sensor is known as a bipolar sensor. The sensor also has built in notch and band-pass filtering to try and remove any noise. The notch filter is there to remove any mains noise that could be coming off of any nearby devices. This would be set to 50Hz here in the UK, while in other countries it could also be set to 60Hz. The band-pass filter allows frequencies starting from 90Hz up to 500Hz to pass. The reasoning behind this is that the range for most myoelectric signals is between 0-500Hz. Although we will be cutting off some of the lower end frequencies with both the notch and band-pass filters, previous studies have shown that the dominant frequencies for myoelectric signals is between 50-150Hz.

3.4 Upper Limb Anatomy

Understanding the muscle and bony structure of the forearm is essential to properly operating and controlling a myoelectric prosthesis. Not including the hand, the forearm has two bones going from the elbow joint to the wrist. These are the ulna and the radius. Much of the muscles in the forearm are used for positioning and manipulating the fingers and wrist. The remaining muscles of the forearm act to flex the elbow joint, along with pronation and supination of the forearm. In total, there are nineteen muscles in the forearm, not including any located solely in the hand. The muscles of the forearm are organised into two compartments, anterior and posterior. The muscles we will be focusing on are shown in figures 3.1 and 3.2.

3.4.1 Anterior Compartment of the Forearm

The anterior compartment of the forearm has eight muscles which are further subdivided into the more superficial and deeper muscles. With only one exception, the muscles of the anterior compartment act as flexors of the hand, digits, or elbow. The main muscles that we are concerned with here, since we will be monitoring them for control of the prosthesis, are the flexor carpi radialis (FCR) and the flexor digitorum superficialis (FDS). The FCR is considered to be in the superficial portion of this compartment and is responsible for flexing and radially abducting the hand. The tendon of this muscle inserts into the hand at the bases of the second and third metacarpals and so will have some effect on the index and middle fingers. The main body of the FDS is considered to be in the deep portion of the anterior compartment. However, it is a relatively large muscle and a portion of it is superficial to the skin, which allows us to pick up the EMG activity from it using surface sensors. The FDS acts to flex the wrist, metacarpophalangeal and interphalangeal joints. Thus, it acts on a variety of digits and should be useful when monitoring gestures that rely on finger flexion.

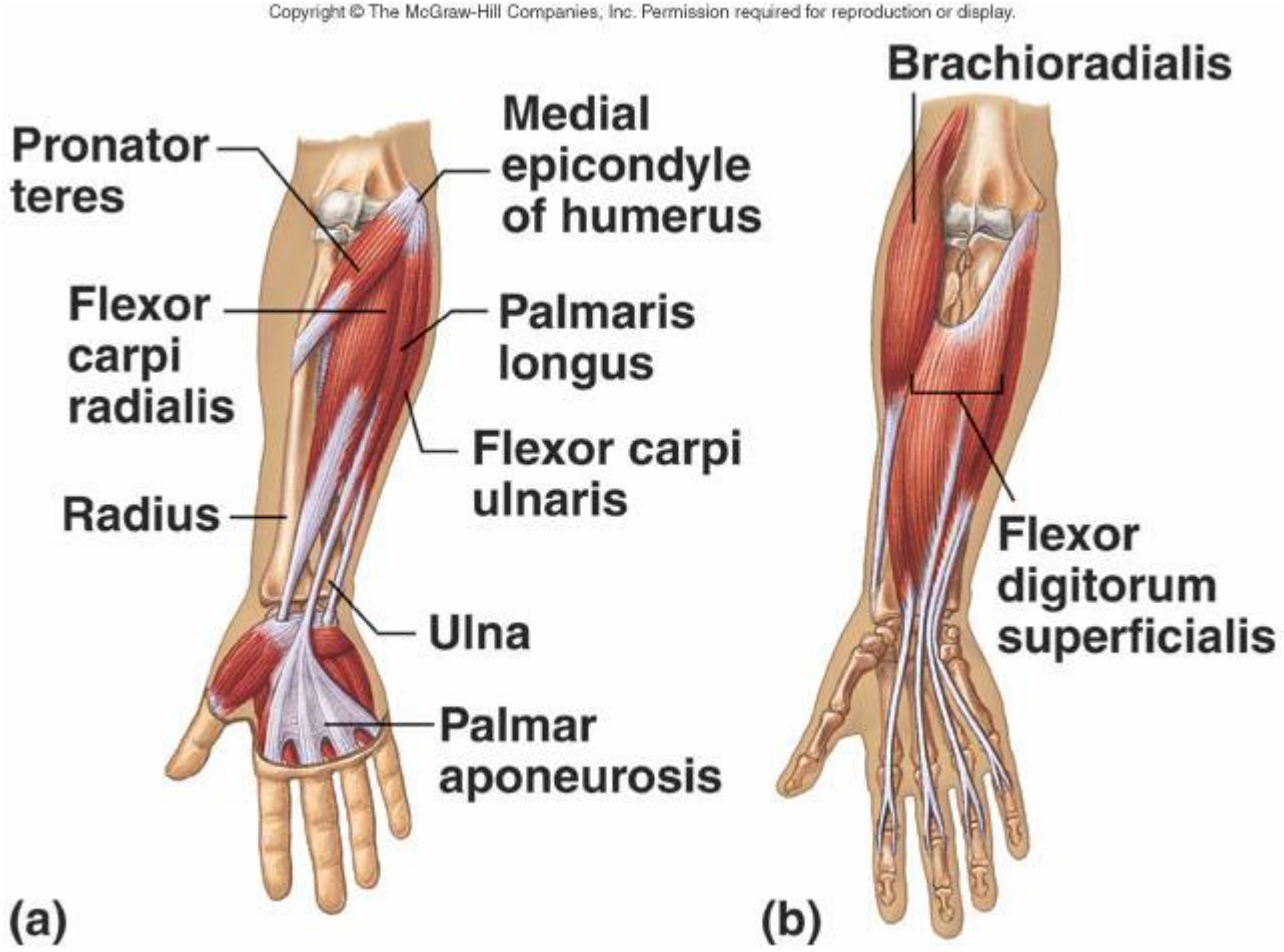

Figure 3.1: FCR is labeled in (a); FDS is labeled in (b) [35]

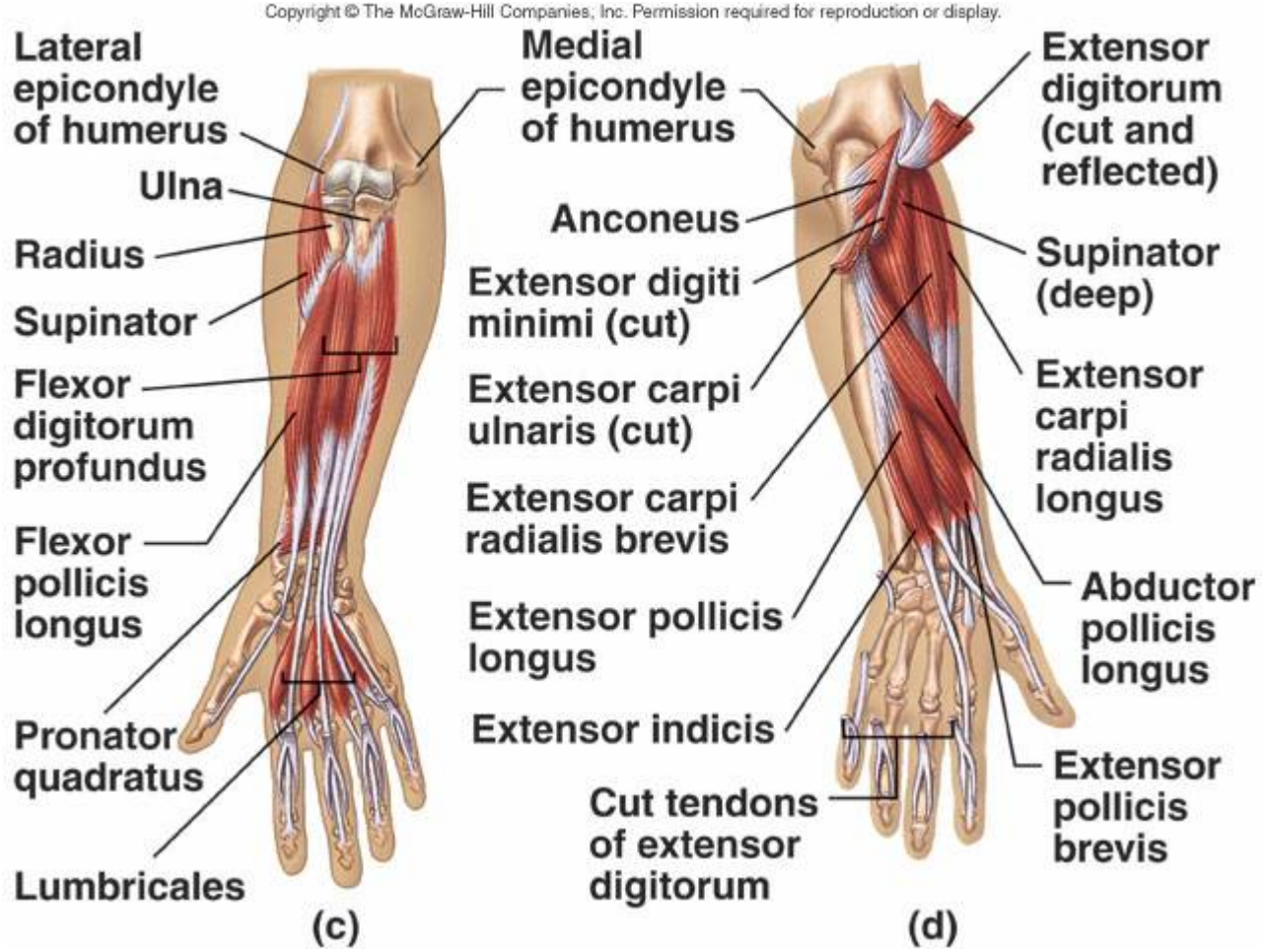

Figure 3.2: ED and ECU are shown in (d) [35]

3.4.2 Posterior Compartment of the Forearm

The posterior compartment has a total of eleven muscles in it. Most of the muscles in this compartment are extensors of the hand, wrist, and digits. Like the anterior compartment, the posterior compartment has muscle considered both superficial and deep. The muscles we will be monitoring here are the extensor digitorum (ED) and extensor carpi ulnaris (ECU). This main function of the ED muscle is extension of the hand and fingers. The end of this muscle divides up into four tendons, known collectively as the extensor indicis, which attach to the fingers. It is also quite superficial to many of the other muscles of this compartment. Because of its position and functions, it is an ideal muscle to monitor for control of the prosthesis. The ECU is located medially to the ED. This muscle is mainly an extensor of the wrist, but its tendon inserts into the hand at the fifth metacarpal joint, and does have some effect on the extension of the fingers.

3.4.3 Optimal Placement of Electrodes

For our purposes, we will be monitoring the flexor carpi radialis, flexor digitorum superficialis, extensor digitorum, and the extensor carpi ulnaris. These four muscles are ideal for our purposes due to a few factors. Our goal is to have the subject perform certain gestures so the prosthesis can imitate them. Thusly, we want to measure the major muscles involved in manipulating the fingers which are also easily located. The fact that they are rather superficial also makes them ideal for sEMG monitoring.

There are several extensor muscles involved in manipulation of the fingers and wrist. However, most of them are specialised for an individual finger or merely act on the wrist. The extensor digitorum is the only muscle of the extensor muscles of the hand that has some influence on all of the fingers, except the thumb. Because of this, we chose this muscle to be the one we will monitor for extensor patterns in the gestures. The electrode would ideally be placed on the belly of the muscle nearer to the elbow where it is most prominent. Since this muscle tends to be rather superficial, it is easy to locate and there should not be too much interference from surrounding muscles. This muscle is innervated by the radial nerve. [17]

The extensor carpi ulnaris is mainly a wrist extensor, but has shown some effect on finger extension while testing. The main reason for choosing this muscle was that it was comparatively large and superficial, which would make collecting sEMG readings easy. Other muscles, such as the extensor indicis or extensor digiti minimi, control more of the fingers than the ECU but are located too deeply for sEMG data acquisition. The [17]

The flexor carpi radialis helps with both wrist and digit flexion, and is ideally located opposite the extensor digitorum. It is innervated by the median nerve. However, since this muscle is somewhat smaller and more closely surrounded by other muscles, interference is more of a concern here than with the extensor digitorum and can be somewhat more difficult to locate. The tendon closest to the thumb extends off of this muscle, so bringing the wrist into flexion will cause the tendon to become more prominent and thusly easier to identify. The optimal placement of the sensor would be one third of the distance away from the elbow towards the wrist following along the path of the tendon. [17]

The flexor digitorum superficialis is an extrinsic muscle with several tendons that come

from it and attach to four of the fingers, excluding the thumb. This muscle flexes these second through fifth fingers, thus making it an ideal muscle for monitoring of gestures relying on finger flexion, such as grip. The placement of the sensor should be proximal to the elbow, along the ulna. [17]

4 Myoelectric Prostheses

4.1 Introduction

Myoelectric prostheses are a unique type of prosthesis that differs from the older cable operated models. These are motorised prosthetics that are able to utilise the myoelectric signal from a subjects residual arm as a way to control the prosthetic. These present a great possibility to restore quality of life and capabilities for amputees.

4.2 Myoelectric Prostheses versus Other Models

Myoelectric prostheses can afford a greater range of capabilities over that of older types. Being able to move each individual finger irrespective of the others allows the prosthesis to be used in a number of situations that could be more difficult with only the older prehensile hooks. They also benefit from a cosmetic point of view. Being much more hand-like in appearance, they may be more appealing to some people. Many also may come with a cosmesis to make it look like a realistic hand. A novel design for a myoelectric prosthesis that has a variable grip strength was tested by Chu et al. wich was capable of various hand gestures. [10]

There are some drawbacks to using a myoelectric hand though. Due to their motorisation they may end up being heavier than traditional prostheses. There is also the possibility that they will not afford the same amount of strength other prostheses are capable of. Having a stronger motor in the prosthesis will likely make it heavier and more bulky, so there is a trade-off there.

One of the major drawbacks is the need for the subject to learn how to operate the prosthesis. With myoelectric prostheses it is not necessarily a straightforward procedure that can be followed. A series of quick contractions of one or more muscles could signal for a certain gesture. The subject would have to learn to perform these contractions to get the prosthesis to respond appropriately, but some people may have difficulty with this if their amputation makes it uncomfortable.

4.3 Touch Bionics' iLIMB

The iLIMB is a myoelectric prosthesis of the forearm designed and manufactured by Touch Bionics, a Scottish company located in Livingston, Scotland. The iLIMB is an advanced, modern prosthesis capable of being positioned into several predetermined hand gestures. The prosthesis is also capable of varying the strength of its grip on an object as the user sees fit. The current model relies on two sEMG electrodes for its control signal from the subject. These electrodes are embedded within the prosthesis socket and so should be correctly positioned on the arm if the socket is fit properly. Currently, certain gestures are signaled to the prosthesis by a series of rapid flexions or extensions of the forearm muscles that the electrodes are monitoring. Capabilities, features, and care of the i-Limb can be

found in the service and fitting manual [4] and the i-Limb data sheet [2]. Pictured below in figure 4.1 is the Touch Bionics iLimb performing a pointing gesture.

Figure 4.1: The Touch Bionics' iLIMB performing a point gesture

5 Processing the EMG Signal

5.1 Introduction

Control of most myoelectric prostheses today rely on a rather simplistic set of rules. The subject will likely have two electrodes located somewhere on their forearm, one monitoring an extensor and the other monitoring a flexor muscle. The prosthesis control software will then have a set of pre-programmed gestures and movements that the subject can perform. They accomplish this by providing a series of quick contractions of either muscle in certain patterns. For instance, two quick contractions of the flexor muscle could signal for the hand to point, or a prolonged contraction of the extensor muscle could signal for the hand to open.

While this is a straightforward approach to controlling the prosthesis, it is not the most intuitive. The patterns of muscle contraction can have no obvious relation to the actual gesture being performed and so add an extra burden to the subject when learning how to operate the prosthesis. This method can also have a limiting effect on the amount of actions the prosthesis can perform, due to the pattern of muscle contractions the subject would have to do. In order to deal with the limitations of traditional methods of prosthesis control, there have been a number of other techniques that have been looked in to that would try to make operating these prostheses more intuitive while at the same time making them more versatile. One such is using pattern recognition of the sEMG signals to properly identify these gestures. Some research has already been done, looking at identifying several different hand gestures using a series of sEMG sensors. [6] Others, such as Shuman [30] have shown it is possible to correctly identify with a high accuracy hand gestures using sEMG electrodes.

5.2 Techniques for Processing

When analysing the EMG data that has been collected, we can look at a number of things that could help us identify which gesture the subject is performing. For instance, we could look at the amplitude of the signal. We could also look at the frequencies present in the signal or the number of times the signal crosses the zero. What we are looking for are any defining characteristics to the EMG signals that would allow us to easily distinguish between the different gestures.

It takes two steps to analyse an EMG signal. Firstly we need to extract features from the signal. Feature extraction is a form of dimensionality reduction, the process of reducing random variables in a set. EMG data is rather random and also is expected to be redundant, so reducing the signal to a features vector is necessary to obtain an adequate description of it. Features that can be extracted are mean absolute value, zero crossings, autoregressive model coefficients, and others. As Khokhar et al. [14] note, feature extraction basically leaves the signal unusable. Several people have suggested different features that could be used in sEMG classification [32, 15] What needs to be done is to segment the signal before the gesture is performed. They propose a 100 120ms delay to segment the signal and extract features from this segment before the control signal is sent. Secondly we need to perform a classification. Classification is the determination of which category an observation should

be allocated in to. For our purposes, each class would be a different hand gesture that the prosthesis would be performing.

Features that Khokhar et al. [14] extracted from two sets of data were sEMG rms value, autoregressive model coefficients, and waveform length. For their classifier they used support vector machines (SVM). SVM looks at new data that is input into the system and determines which of two classes this data fits in to. It must first be given some training examples so that it can properly identify which category the new data should go to. This is a method of pattern recognition and is the where this research is heading towards. We will be discussing this in more detail later on. A paper written for Delsys, a sEMG sensor manufacturer, has a good overview of typical sEMG data acquisition, filtering, and processing to obtain a working signal. [12]

5.2.1 Autoregressive Modeling

Hefftner et al. [13] review the possibility of using an autoregressive model for control in functional neuromuscular stimulation (FNR). Although FNR deals with stimulating paralysed muscle tissue to retrain movement patterns, this paper focused on analysing EMG data in a way that could also be applied to prosthesis control.

In the paper the authors treated the EMG signal as a stochastic process. By this they mean that it is essentially a random signal. Because of this we cannot describe the signal in terms of ordinary differential equations. An autoregressive model is a mathematical model used to predict the outcome of some natural process based off of past events and data. The autoregressive model makes use of autocorrelation to determine the parameters of the system. Autocorrelation itself helps to identify a commonality embedded in a seemingly white noise signal.

5.2.2 Fourier Transform

The Fourier transform is a very useful tool for signal processing in various situations. A signal is often initially represented as a function of time. Here, the signal is said to be in the *time domain*, where the signal is represented as an amplitude that varies over time. The Fourier transform takes this equation in the time domain and converts it into the *frequency domain*, where the signal is being represented with an equation that gives amplitude as a function of frequency. What this does for us is to view the different frequency components of the signal and identify which ones are more dominant. This technique would be a very useful asset if we can identify certain major frequency components in an EMG signal that are unique to a gesture being performed.

The Fourier transform is derived from Fourier series analysis. The assumption made with using a Fourier series is that any periodic signal can be represented as a series of sine and cosine functions summed. Each trigonometric function represents a different harmonic of the signal, each with its individual amplitude and phase shift. The basic Fourier series is given by equation 5.1

$$x_p(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)] \tag{5.1}$$

$$a_0 \equiv DCterm, a_k \equiv c_k \cos(\phi_k), b_k \equiv -c_k \sin(\phi_k) \tag{5.2}$$

Although this is a useful way to represent the signal it can get cluttered, so using trigonometric identities we can change this equation series form into a summation of a series of complex exponentials which will keep it more compact. The signal can then be represented thusly in equation 5.3,

$$x_p(t) = \sum_{k=-\infty}^{\infty} X_k e^{jk\omega_0 t} \tag{5.3}$$

$$X_k = \frac{1}{T_0} \int_{\frac{-T_0}{2}}^{\frac{T_0}{2}} x_p(t) e^{-jk\omega_0 t}\, dt \tag{5.4}$$

For various different periodic signals, X_k can be simplified depending on which type of signal it is. This form of the equation provides the basis for the Fourier transform because although this is a better form for describing the signal, it is not so useful for non-periodic signals. Instead, we assume the period of the signal is at infinity and so take the limit of the formula for X_k as $T \to \infty$ and get the following equation 5.5

$$\mathcal{F}[x_p(t)] \equiv X(\omega) = \int_{-\infty}^{\infty} x_p(t) e^{-j\omega t}\, dt \tag{5.5}$$

This is the continuous form of the Fourier transform and although useful in a theoretical sense, we cannot use it in a practical sense. Our EMG signals will be recorded and converted into a digital format. This means our signal will be converted into a discrete form, eliminating the possibility for us to use the continuous form of the Fourier transform. Fortunately, we have the use of the discrete Fourier transform (DFT) that can be employed, given in 5.6

$$\mathcal{F}[x_n] \equiv X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi \frac{k}{N} n} \tag{5.6}$$

where N is out number of samples. This is the form of the Fourier transform most useful to us, and it will be able to rather accurately represent the continuous time transform of our original signal. We will of course be ustilising MATLAB to calculate these transforms and as such will take advantage of the fast Fourier transform (FFT). This will provide the same results as the DFT, but will make use of the Cooley-Tukey algorithm to break down the Fourier transform into several smaller transforms to reduce computation times.

5.2.3 Power Spectral Density

The power spectral density (PSD) of a signal describes how the power of the signal varies with frequency. For us to ustilise the PSD of a signal, we must first assume that it is a power signal. That is, we assume it is a wide-sense stationary (WSS) signal, where the power of the signal does not decay over time, as long as the signal exists. Otherwise, the PSD of the signal would not exist. The power of the signal being described here is analgous to the power that would be dissipated in a resistor with a voltage difference applied across it. This

is defined as the signal value squared, which gives us the total average instantaneous power, as shown in equation 5.7.

$$P(t) = x(t)^2 \tag{5.7}$$

for a signal x(t).A little more useful to us is the normalised mean power of the signal, which is given in equation 5.8

$$P_m(t) = \lim_{T\to\infty}[\frac{1}{2T}\int_{-T}^{T} x(t)^2\,dt\ ] \tag{5.8}$$

What we would really like to look at is the distribution of this power over the signal frequencies. Assuming the power density spectrum is given as $s_x(\omega)$, we know that the total area under the spectrum is the average power of the signal. Thusly, we can also give the average power as equation 5.9

$$P(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} s_x(\omega)\,d\omega \tag{5.9}$$

We cannot take the Fourier transform of the signal due to random process signals being not absolutely integrable. However, according to the Wiener-Khinchin theorem, the PSD of a WSS signal can be given by the Fourier transform of that signals' autocorrelation function. Since the EMG signal is a continuous-time signal, we use the continuous form of the autocorrelation function, given as equation 5.10

$$R_{xx}(\tau) = E[x(t)x^*(t-\tau)] \tag{5.10}$$

where E() represents the expectation function, which gives us the first moment of the signal, τ is a time delay, and $x^*(t)$ is the complex conjugate of the signal. We can then take the Fourier transform of the autocorrelation function and obtain the signal PSD, as given in equation 5.11.

$$s_x(\omega) = \int_{-\infty}^{\infty} R_{xx}e^{-2\pi i f\tau}\,d\tau \tag{5.11}$$

5.2.4 Time-Frequency Analysis

Plotting a spectrogram may be of more use to us visually in order to see how exactly the signal is behaving. A spectrogram basically shows the power spectral density of the signal as it varies over time. This signal processing technique is often used in sound identification applications. We usually have time as our x-axis and frequency along the y-axis. Although it is possible to have amplitude along the z-axis forming a three-dimensional plot, it is generally more useful for visual representations to have the amplitude plotted using a colour axis. Generally a spectrogram is calculated using the time-domain data and applying the short-time Fourier transform. It basically shows us in what frequency range the power is mainly located at specific times.

5.3 Pattern Recognition

The ultimate goal of this research is to show the viability of utilising pattern recognition techniques in controlling the prosthesis from the EMG readings. Modern pattern recognition

techniques can prove to be quite useful in identifying unique inputs, known as classes. Generally, this is done by taking a large body of sampled data of all the classes being considered and extracting useful features from each piece of data which should be unique to the class that piece of data belongs to. The data is then often preprocessed condition the data by removing outliers, normalising the range, and other methods to clean the data. We then need to extract the relevant features that we want to be using.[33]

Feature extraction and selection will generally result in a dimensional reduction. [27] What we want is to be able to take as few features as possible that stand out in particular for each class, and the combination of those features should be unique to each class. Keeping the number of features being considered to a minimum will reduce computational requirements, thus reducing the computational time. Generally, features that stand out and are unique to that certain class will be used as the dimensions being considered as features. Average power, maximum power, maximum amplitude or others may all be used as features. These features then should to be processed in some way before we can build a classifier, in order to remove outliers or better prepare the data for classification purposes. Principal component analysis (PCA) is a useful method of highlighting patterns in data sets of higher dimensional value, as well as data reduction. It is often used in image compression since it is possible to remove some dimensions of the data set without much loss of information. For purposes such as sEMG gesture recognition, it could be useful in emphasizing certain trends in the signal. There are other techniques available as well, but we will leave these alone for this study.

A classifier is an algorithm that is employed to assign input data a classification. Before the classifier can be used however, it must be trained using a set of data where the classes are already known, so that future data that is input into the classifier will have something to compare to. Training the classifier will basically create a unique set of processes based on the known data. When unknown data is run into the trained classifier, the classifier will process this data and assign this data a class. Increasing the amount of data where the class is already known for training the classifier will likely improve its accuracy. We will likely be using a binary classifier so that we will be comparing each class at a time, rather than all classes together at once. This way we can get a better look at what rate each class is being misidentified as another class, and so forth. The results of a binary classifier for each input will be one of four. If the prediction of the classifier matches the actual class, it is said to be a *true positive.* However, if the predicted outcome does not match the true class of the input, this outcome is labeled a *false positive.* Likewise, when the classifier correctly predicts the class is not the main class we are looking at it is called a *true negative.*, and a misidentified negative as a positive is a *false negative.* Once all of the predictions of the classifier are labeled, we can do several things with them to show how effective our classifier is behaving.

We can now use a few methods to illustrate the effectiveness of our classifier in correctly identifying new input data classes. The simplest would be to plot a confusion matrix. The confusion matrix is a 2x2 graphical matrix, with the columns representing the true classes of the data while the rows represent the predicted classes. The percentage of positives that were correctly identified (true positives) should be placed in the first cell of the matrix while our true negatives should be placed in the second row second column cell. From the confusion matrix, we will be able to calculate the accuracy of the classifier for this particular class.

Using an ideal classifier, our true positive and true negative values should be 100% without any false positives or negatives, giving us 100% accuracy. In real circumstances of course, there will likely be at least some misidentification.

One other such method is to plot the receiver operating characteristic (ROC) curve. This takes a binary classifier and shows a plot of the true positive rate on the y-axis, and the false positive rate on the x-axis. When considering the ROC curve, the true positive rate could also be considered equivalent to the sensitivity of the classifier and the false positive rate is equivalent to 1 - specificity. The ROC curve basically shows the benefits (true positives) versus the costs (false positives). A single confusion matrix would be plotted as a single point in ROC space. Ideally, we would have a classifier the produced a point in ROC space at (0,1). This indicates zero false positives and 100% true positives. Oftentimes, they are also considered to be normally distributed. Moving the threshold of the Gaussian curves of the true positives and true negatives will allow us to build a curve in ROC space which will hopefully better illustrate the classifier functionality. When this curve is plotted, we can do so for the remaining classes as well. Doing this will help to compare the capability of the classifier for each class. Likewise, we may also want to build several ROC curves using different classifier algorithms for the same class to see which works best.

5.3.1 Classification Errors

Li et al. [18] got an average rate of classification error of about 6% when they used six classifications in their model. However, they also tested using 11 classes which resulted in an average error rate of about 21%. In addition, the average error for subjects with transradial amputations was 31% when using the 11 motion classes. Reducing the classes to six significantly reduced classification error to 7% for those with transradial amputations and brought it more closer in line with the overall average. The discrepancy in classification error between the subjects with intact arms and subjects with a transradial amputation showed that in some cases it may not be beneficial to be using subjects with intact arms for improving prosthesis control algorithms. Unfortunately, they did not mention the methods they used for feature extraction and classification, which could have some effect on their accuracy.

Khokhar et al.[14] looked at two cases, one where they defined 13 classes and another where they had 19 classes. The set where they used 19 classes had an average accuracy of 88%, while the set with 13 classes had an accuracy of 96%. As stated earlier, they extracted several features and used SVM as their classifier. It should be noted that their classes where mainly limited to varying intensities of wrist extension and flexion, and ulnar and radial deviations, so the actual muscle contractions are not that different between classes which may help in identification.

The segmentation window comes in to play with classification errors here as well. Longer windows will provide the system with a greater amount of data and should make it easier to classify the data. However, this means a longer delay for the subject before the control signal is sent to the prosthesis. If the segmentation window is too long, it may make operating the prosthesis unsatisfactory for the user. Smith et al. [31] showed that the optimal window length should be between 150 250ms to have adequate control delay without sacrificing classification accuracy too much. However, techniques have been proposed to more adequately

detect the onset of muscle contraction [16], which could reduce detection times.

5.4 Number of Electrode Channels

For our experiment, we plan to look at using four electrode channels for controlling the prosthesis. Many current myoelectric prostheses use only two electrodes, and while this works well for the current system of gesture control, it would be unlikely that only two sensors could accurately identify the correct gesture being performed. a number of previous studies have used four to six electrodes and obtained low instances of classification error in both healthy subjects and subjects with amputations. Li et al. [18] used six electrode channels in their analysis. Although they started off using twelve channels, their study showed that using more than six failed to produce any benefits in reducing classification errors. They recommended using 4 - 6 channels for optimal readings. Unfortunately, having a high number of channels increases the complexity, and thus the cost, of the prosthesis. The sensors alone are somewhat costly [3], since they need to be robust enough to last a long while without the need for replacement. It also will add to the weight, which can be an extra burden to the subject using it. Since the current recommendation is 4 - 6 electrode channels, we will be looking at using four electrodes for the pattern recognition control.

5.5 Sampling Frequency and High-Pass Cut-Off

It has been shown that the usable information in EMG signals is present between 0 1000Hz, and so choosing an adequate sampling frequency that satisfies the Nyquist criterion should be chosen. If following Nyquist the sampling frequency should be 2kHz, twice the highest frequency present. However, there are other factors that come into choosing an appropriate sampling frequency that might not necessarily follow the Nyquist criterion exactly. According to Hefftner et al. [13] the main power of the EMG signal lies in the 300Hz range. In addition there is the problem they run in to when utilising an autoregressive model the low end frequency that will be present is dependant on the model order, following the equation 1/pT, where T is the sampling interval and p is the model order. Their model was a fourth-order model, so the minimum frequency that would present itself if following Nyquist would be 500Hz. Anything below that would be attenuated, which is a problem considering that the main power of the signal lies in the 300Hz range. As a consequence of this, they suggest using a sampling frequency of 500Hz if using a fourth-order autoregressive model.

Supporting Hefftners claim, Li et al. [19] showed that using a 500Hz sampling rate showed a classification accuracy decrease of only 2% compared to a 1kHz sampling frequency. This same paper was also able to show that having a high-pass cut-off frequency of 60Hz decrease classification accuracy by only 0.1% compared to a 5Hz cut-off frequency. This is quite important since as they note in the paper, motion artefacts occur in the 1 50Hz range.

6 Methodology

6.1 Overview

The basic methodology of this research was to obtain the sEMG signal data from several subjects, and then process this data using software such as MATLAB in order to see if any of the signals had uniques characteristics to them.

6.2 Test Equipment

6.2.1 Touch Bionic's sEMG Sensor

There were several pieces of equipment needed in obtaining the EMG data from the subjects. The Touch Bionics EMG sensors that are used for control of the iLIMB were utilised to read the myoelectric signals from the subject. The sensors utilised a differential amplifier as the sensing circuitry. The sensors had three metal electrodes that made contact with the skin. Two of these went to the amplifier while the third was used as a ground element. The particular sensors we used were modified from the commercial versions to remove any rectifying circuitry, and so provided only a raw EMG signal. These sensors have an output voltage offset of 1.2V. The sensors have three connectors; one connection to power them, since they are active sensors and the amplifier needs a voltage source, and one for the ground line. The third connection provided the data output. These connections went to a custom circuit that ensured the sensors were properly powered at 7.4V and were protected from possible over voltages. [5]

The data connection went to the analogue-to-digital conversion ports on a Micro 1401 data acquisition unit. Each sensor utilised separate ADC channels on the 1401. The subject was situated in front of a computer running the MATLAB code for the experiment. This computer output the triggering information for when the code tells the subject to perform a certain gesture on port LPT1 to the 1401. The 1401 then output both the triggering data and the EMG data via USB to a second computer. This second computer records the data from the 1401 using the Spike 2 signal acquisition software.

6.2.2 The Micro 1401 ADC

The Micro 1401 analogue to digital converter is an ADC specialised for use in research situations and is fast and fairly easy to use. The standard model has four ADC input channels, two DAC output channels, a trigger port, two event inputs, two digital outputs, and an external clock port. In addition, it has several LPT ports that can be used for external triggering. The unit is also expandable, allowing it to accommodate up to 64 input channels. It outputs the recorded data to an external computer running the associated Spike2 recording software via USB. For this experiment, we used the standard four ADC inputs for the EMG data acquisition and the LPT digital input port for the digital markers coming from the test computer.

6.2.3 Overvoltage Protection

The typical output of the sEMG sensors should not exceed ±5V. However, we want to take precautions to prevent any abnormalities where a voltage higher than that could occur. Too high of an input voltage to the Micro 1401 could damage the ADC. The circuit employed was a simple zener diode voltage regulator circuit to cut off any signal voltage that would exceed ±5V.

Figure 6.1 depicts the test setup as was used, and figure 6.2 shows a diagram of the circuit used. This is just for one channel; the actual breadboard used had a separate iteration of this for each channel

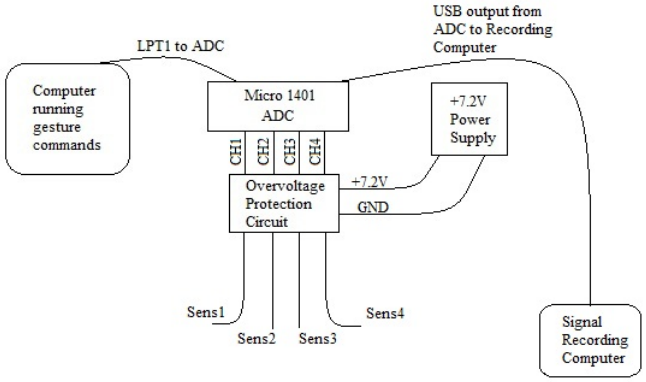

Figure 6.1: Experimental Setup

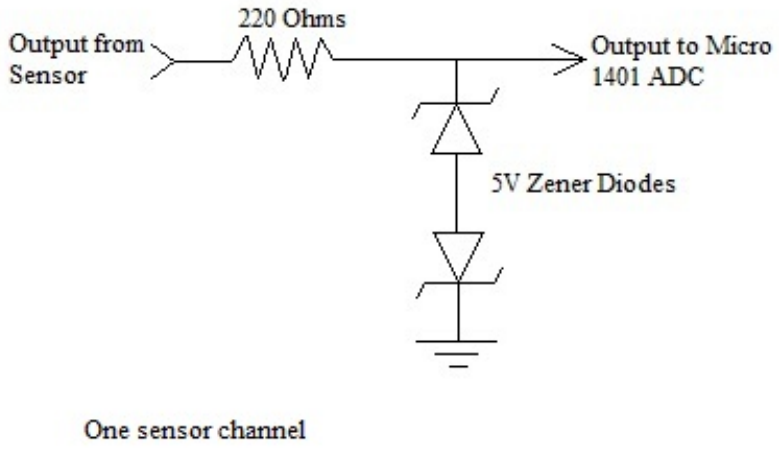

Figure 6.2: Voltage regulator circuit diagram. Two simple Zener diodes to prevent voltages exceeding + or -5V

6.2.4 Software in Data Acquisition and Processing

Spike 2

The Spike2 data acquisition software was designed for use with CED products and as such easily interfaces with the Micro 1401. The software initialises the Micro 1401 and tells it which channels we will be monitoring. The software controls all of the settings for the Micro 1401, including the sampling frequency, ADC resolution, and others. It is also capable of some filtering and analysis of the data. We will be recording all of our raw data using this software, and then exporting it into MATLAB file formats.

MATLAB

MATLAB is a numerical computing program developed by MathWorks. It is capable of a wide array of matrix manipulations, data manipulations, applications of algorithms, and more. The capabilities of MATLAB are wide enough that it is used in a large number of sectors, ranging from science, engineering, economics, and others. MATLAB executes functions according to its own proprietary programming language. Because of this, commands used to implement various functions or actions can be stored in a script file. We can have a script file that runs several consecutive commands in order to automate long or tedious tasks, reducing the possibility of user error and saving on time that would be otherwise taken up by entering commands. In addition to the basic functions that MATLAB is capable of, its functionality can be expanded on by installing separate toolboxes which generally have some specific purpose to their included functions. We will be using the Signal Processing Toolbox developed by MathWorks, along with the Pattern Recognition Toolbox that was developed by New Folder Consulting, a company that develops tools for use in pattern recognition applications.[34]

The Signal Processing Toolbox will allow us to easily calculate the fast fourier transform, power spectral density, and spectrograms of the signals. We will also be able to perform some more advanced analysis, such as wavelet transforms. The calculation of periodograms of the power spectral density with the signal processing toolbox will also make it easier to obtain the average power for each signal. The pattern recognition toolbox will allow us to use our sampled data and train classifiers. It will allow us to test these classifiers against our sampled data and to easily build confusion matrices and ROC curves.

6.3 Subject Preparation

Before we started any data acquisition, we needed to ensure that our subject had normal functioning in their forearm musculature and that they had good or corrected eyesight. It was also important to know whether the subject had any skin allergies, since we would be using an abrasive gel to clean the area of skin where the sensors were coming into contact with. In addition, we made sure there were no other limiting factors for the subject while completing the procedure.

Once that was sorted we could begin preparing the subject for the experiment. We first needed to locate the appropriate locations for the sensor placements. We begun by locating

the extensor digitorum. Due to this muscles superficiality and size, it was not be too difficult to locate. The subject was first asked to move their middle finger up and down. Muscles in the anterior area on the forearm near the elbow would visibly start to contract. This was a rough location of the extensor digitorum, but we wanted to be more precise to the centre of the muscle. We next held the subjects arm at the hand and asked them to push upwards. This caused the muscle to contract isometrically and would be noticeably tense. Without letting the subjects arm move the experimenter then felt for the muscle and made a mark over the centre. This was the location of the first sensor.

Next we needed to locate the flexor carpi radialis. This was be a little more difficult due to the muscles smaller size and positioning amongst other muscles. To locate the muscle we had the subject flex their wrist while they had their palms facing upwards. The researcher then applied an opposing force while the subject tried to flex their wrist. The tendon furthest lateral at the wrist would be coming off of the muscle we are looking for, so performing this gesture would make this more prominent. We followed this tendon towards the elbow angled medially while continuing the opposing force. Alternating applying the force and relaxing helped to locate the muscle. Ideally, it would be about one third of the way away from the elbow towards the wrist on the medial side on the posterior side of the forearm.

Once the proper locations for the sensors had been located, we prepared the skin surface to optimise it for maximum conductivity. We swabbed the area with an abrasive gel to clean the area and reduce skin impedance. We then placed the sensors with their contacts touching the skin and use some tape to hold them. Once they were in place, we used an additional band of fabric to hold them in place. Any movement of the sensors across the skin will generate artefacts and false data, so we needed to be sure the sensors were secure. Once the sensors were in place and secured, we used an additional cloth wrapped around the wrist to restrain any wrist movements. Since these muscles are both involved in wrist extension or flexion, movement of the wrist will generate myoelectric signals. We wanted to only be recording the gesture signals, so preventing wrist movement was very important.

Now that the sensors were correctly attached and the subject was comfortable, they needed to be situated in front of the computer that will be displaying the experiment gestures. The arm would be resting on a table so that the subject is not lifting it. Once the subject acknowledged they can see the computer screen and reported no discomfort, the experiment begun.

6.4 EMG Data Acquisition

The computer the subject was viewing was running a MATLAB program that displayed a random gesture that the subject would imitate themselves for the duration that it is displayed on the computer. There were four gestures that the subject performed which were a precise pinch, lateral key grip, index point, and tripod. The program started off by giving the subject a few seconds for rest. It then informed the subject of the first gesture to be performed a few seconds before the gesture should actually be performed. This way the subject knew ahead of time what will be performed. The program then told the subject to perform the gesture and hold that position for five seconds. After, it then gave the subject a few seconds of rest while informing of the next gesture to be performed.

Figure 6.3: Grip Gesture

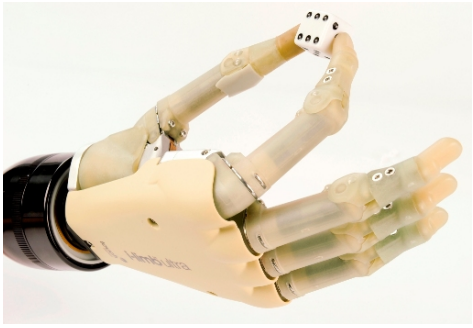

Figure 6.4: Pinch Gesture

Figure 6.5: Point Gesture

Figure 6.6: Tripod Gesture

Each trial run followed this pattern five times, choosing a random gesture each time. We repeated the trial thirty times to get a total of 150 gestures recorded. In addition, we asked the subject to periodically perform the gesture at their maximum strength so that we could see what to expect for their peak EMG readings. We ended up with three recordings of maximum contractions for each gesture. For the maximum contracture gestures, we asked the subject to gradually build up to their maximum rather than going straight into it. Once they reached their maximum, we asked them to hold this for five seconds before releasing. This procedure is rather straightforward and did not cause any discomfort for the subject.

All the while that the subject was performing the gestures, we were recording the output data from the 1401 using the Spike2 software. The output would show a nearly flat line for each channel when the subject was at rest and a rather busy one when performing a gesture. When recording the data, the Spike2 software saved the raw sampled data as an .SMR file. The configuration is set up so that the file contained six channels of information. Four of the channels were the four EMG data channels from the sensors. The probes being used to connect the Micro 1401 to the sensor outputs have a 10x scale, so this was set in the configuration file. The remaining two channels were the digital markers. One channel of the

digital markers recorded any keystrokes made during the recording session. This was used primarily for determining the start and end of the maximum contraction gestures. The other digital marker channel recorded the output from the LPT1 port of the computer running the gesture MATLAB experiment. The MATLAB script output a code to the Micro 1401 whenever a gesture was to be performed, and then indicated when the gesture should stop being performed. The code is specific to each gesture so that it would be easy to identify the gesture being performed. This also would aid us in determining the start and stop times of the gesture, which made it easy to set up an automated script that will epoch each gesture. Once all trials were done, we were then finished recording our data from the subject.

6.5 Data Processing

Before we could start processing any of the recorded EMG data, we needed to convert the file into a format usable by MATLAB. The newest update of the Spike2 software allowed us to easily export the existing files into a MATLAB variable file. We needed to make sure that while exporting the data we ensure the times being exported were from the beginning of recording to the MaxTime() recorded. We also needed to make sure that the digital markers have their labels exported and that each channel would retain its name that was set in the configuration. If all the settings were correct during the exporting process, the output file should contain all four EMG data channels that were recorded, labeled for example as "e1_FCR" for the FCR muscle channel if the original file had been saved as "e1.smr". If the file being exported was a regular trial, only the digital marker channel that was the output of the LPT1 port should be present. Likewise, for the maximum contracture gesture files only the keyboard digital marker output channel should be present. The pertinent information from each sensor channel data was mainly the times and values. With the digital marker channels, the main useful information were the times that were recorded.Once we determined that the file was exported properly, we then begun processing the data. Our first step was to epoch each gesture.

6.5.1 Gesture Epoching

Each trial consisted of five different gestures in a row. What we wanted is to look at each gesture individually though, so we needed a process to divide up each trial so that we could focus on each one. While it was possible to go through each trial and do this manually, we had the capability to automate this process and save a significant amount of time and reduce possible human error in the process. In addition, the epoching process eliminated any offset present in each sensor, any consistent noise on each channel, and truncated the first 1.5 seconds and the last 0.5 seconds of each gesture. This truncation helped to remove any inconsistencies that may have been present at the beginning due to the subject still forming the gesture, or at the end in case the subject began to tire of the gesture.

The MATLAB script used to epoch the data will be taking the data and putting it into a new matrix. To run the script, we must include in the arguments the variable names for each channel used in the trial, including the digital marker channel. In addition, we need to tell it the time for which to determine the base line. It works by first looking at the digital

marker channel and identifying the time of the first gesture. The time for each gesture will be located in the even-numbered rows of the digital marker matrix. The odd-numbered rows correspond to the times when the subject is instructed to rest, both at the very beginning of the trial and at the end of each gesture. The script then multiplies the time of the gesture by the sampling frequency. This is done to determine the actual data sample where the gesture begins at. Since our sampling frequency was at 2000 Hz, the time would be mulitplied by 2000. Since each gesture was held for five seconds, we would expect each gesture would take up 10,000 samples of data. To truncate the first 1.5 seconds from the data, it determines that it will only bring data over to the epoched data matrix 3000 samples after the start of the gesture. It then determines the end of the data to be brought over should be 6000 samples later. This leaves off the last 1000 samples of data, or 0.5 seconds, from the gesture. It proceeds to do this for each gesture and each channel, and stores it in a local matrix that contains all four channels of data for that gesture.

Once each gesture has been epoched, the script determines the baseline which should be subtracted from the data. The script takes the signal starting from the time specified in the input arguments and lasting for 6000 samples, and subtracts this from the rest of the data. This removes any offset that may be present in the signal along with any persistent noise. All of the sensors had some offset, either of 1.2V or 1.5V, so removing the offset was an important step. Once all of this is done, each epoched and corrected gesture signal is put into individual variables, and then these are saved as an external file to the hard drive.

6.5.2 Fourier Transform

One of the first processing scripts that we run is the Fourier transform script. This script will take the Fourier transform of all four channels of the specified gesture and output a plot of each channel.

When running the script, we need to specify in the function arguments the file we are analysing, the gesture that was being performed, whether or not this is a maximum contraction, a filename, and the location to save the plot output to. To indicate which gesture is being performed, we only need to put in the number output to the digital marker channel. To indicate that it is a maximum contraction, we set this value to one. If it is a regular trial being analysed, we can put any other value here.

6.5.3 Power Spectral Density

The power spectral density gives us the power distribution across the different frequency components of each signal. The values obtained here will also be of value later on when creating a classifier for the pattern recognition algorithms. Since we are using real data, we will need to get the spectral estimation of the signal through a periodogram. We start off by defining the window we will be using for our periodogram. A hamming window is fairly standard and gives us a good result, so we will be using that. Just like with the fourier transform, we need to specify that our sampling frequency was at 2000 Hz. We then create an options object in the workspace for each channel of data, which will confer all the specified parameters we want the periodogram to follow. When the power spectral density (psd) function is run, this object will specify that it should be two-sided using sampling frequency

of 2000 Hz. Once that is completed we can input our arguments into the psd function, along with the options object, and we will obtain the periodograms for each channel of data which can then be plotted. Our output values will be power per frequency (dB/kHz) on the y-axis, and frequency (kHz) for the x-axis. Same as with the fourier transforms, to better visualise the entire gesture we plotted all four channels of data in a single figure.

Although the plots of the power spectral densities are useful from a visual perspective, the PSD will also be helpful for a DSP when controlling a prosthesis. Specifically, we can calculate the average power of the signal on each channel which may be used as the value a pattern recognition classifier could be looking at. Obtaining the average power of a signal is simple once the power spectral density has been computed. We merely need to use the 'avgpower()' function in MATLAB for each channel, and then convert that value back into dB using the formula

$$10 * log_{10}(\frac{Power_{freqdom}}{2}) \tag{6.1}$$

These values are stored for later use when we will be calculating the pattern recognition classifier.

6.5.4 Spectrogram Time-Frequency Analysis

The spectrogram of the data gives us the power across the different frequency components over time. In a sense, it is looking at the power spectral density of the signal over time. The spectrogram function in MATLAB produces a three-dimensional image which resembles a topographical map, showing peaks and valleys with the power being represented is higher at the peaks than the valleys. However, it is easier to convey the pertinent information in a two-dimensional representation. What the script does then is to colour-code the plot, with the redder areas being areas of higher power and the blue areas representing lower power. It then looks at an aerial view of the plot so that the x-axis corresponds to time and the y-axis corresponds to frequency. There will also be a colour axis plotted alongside to indicate the range of power being represented.

6.5.5 Other Signal Transforms for Feature Extraction

Due to time constraints, we cannot make use of all methods available for looking at features of the signals. There are several other common methods, such as discrete wavelet transforms (DWT), that could be used. There are also newer techniques that could be employed, such as mother wavelet matrix (MWM) analysis, as proposed by Rafiee et al. [29]

6.5.6 Signal Averaging

Although it may be useful to analyse each gesture individually, it would be better to get an overall idea of the expected gesture pattern for each individual instead. We can do this by averaging all of the gestures that a subject performed with each other. For example, we could take the average of all of the grip gesture signal data and run the previous analysis techniques on the result. By averaging the signal, we should get a better overall idea for what a DSP should have stored in it for pattern matching. In addition, it would also be beneficial

to take global averages for each gesture. That is, average each gestures' signal pattern across all subjects. This has the downside of eliminating any possible individual differences, but would also give a possible DSP a more general pattern to work with. A global average may be more useful if we had used many more subjects, but it will be interesting to see the results with the data we have.

6.5.7 Pattern Recognition

In addition to the previous signal processing technique employed to extract useful features from each data set, we also took a rudimentary look in to some basic pattern recognition techniques that could be employed in a DSP for actual control of the prosthesis. While an end product classifier may end up culling features from one of the signal transforms we will be looking at, since actual pattern recognition is somewhat outside the scope of this project we will be using the RMS values of each signal and each gesture mainly as an example of what our data is capable of. We calculated the RMS values for each gesture across all four data channels and then combined these into a single matrix representing the entire gesture data. We then made use of the Pattern Recognition Toolbox. We split the data into two separate data sets, one which will be used for training the classifier and the other for testing it. The reasoning behind this is that we may get overly optimistic results by using the same data set for training and testing the classifier.

We first have to build a target matrix which will be stored in the dataSet object. The script we are running will first count how many times each gesture performed and then fill our target matrix with that number of labels for that gesture. For example, if there were forty-two grip gestures performed, there would be forty-two rows in the target matrix with the first cell filled with the value of two, our label for the grip gesture. The target matrix holds only one column and the label for each data point. Since we are splitting the data into two sets, this process is repeated twice so that we get a target matrix for each data set.

We then combine all of the data values into a matrix, one for testing and one for training. Once this is done we use the Pattern Recognition Toolbox to create data set objects for each set. This will take into account both the data matrices and the target matrices. The data set object is what the classifier will then be using to train and test the data. Before that though we can plot the data as a scatter to see the grouping and if any gestures will be spread out or if they have distinct regions that they occupy. This is essential, since if there is a lot of spread for the gestures it may make it more difficult for the classifier to work properly.

Next, we will be using the K-next-neighbour (KNN) classifier. We start by generating a classifier object as a KNN and then training it using our training data set. This will give us our fully useful classifier. Before moving on, we can plot the trained data with a coloured overlay with the regions that the classifier is identifying as where each separate class will be located.

We can now test how well the classifier is working. Common tests for this are the confusion matrix or the ROC curve. The Pattern Recognition Toolbox is capable of doing both, which will be our last step in evaluating the classifier.

7 Results

Our subject pool consisted of four female subjects and three male subjects, all in good health with normal or corrected vision and nor reported muscle abnormalities or any other known impediments to the testing process. All test subjects ranged in age between 23 and 28 years of age. We used four sensors on all of these subjects, but for one of them two of the sensors were placed orthogonally on the muscle. This resulted in poor data acquisition and as such her data was not used. All subjects completed the entire test process of thirty trials of five gestures along with three repetitions of maximum contractures for each gesture.

7.1 Raw EMG Data

The raw EMG data showed some differences between the gestures. Some data looked to be noisy, usually a result of the sensor not making full contact with the skin of the subject, In these cases the band securing the sensor to the arm was usually repositioned to ensure better sensor contact with the skin. The figure shown shows several different gestures. The

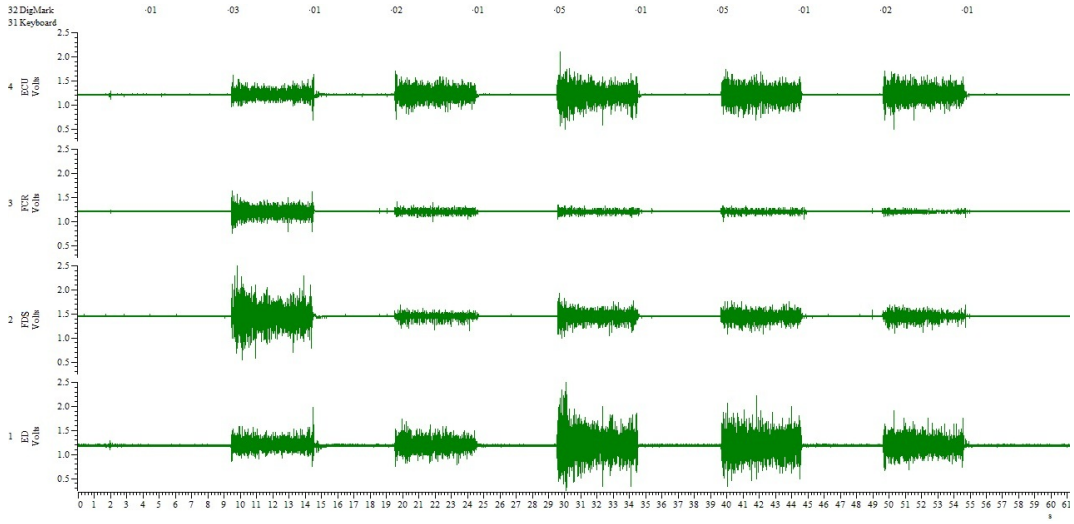

Figure 7.1: Trial 8 from male test subject. Vertical axis indicates voltage for each sensor channel. Horizontal axis shows time in seconds. Digital markers along the top are the gesture codes. 01 - rest, 02 - Pinch, 03 - Grip, 04 - Point, 05 - Tripod

first is a grip gesture and the second a point gesture. The third and fourth gestures were both tripods, while the fifth and last gesture was a point again. All four sensors channels are shown, with the ECU being the top one, the FCR the second one, FDS the third one, and ED being the bottom one. The difference between rest states and muscle contraction are readily noticeable, and it may even be possible to tell each individual gesture from this picture. This, however, was a rather good and clear example. Some other results taken lacked the definition that this one does, and better processing techniques would be needed to identify them.

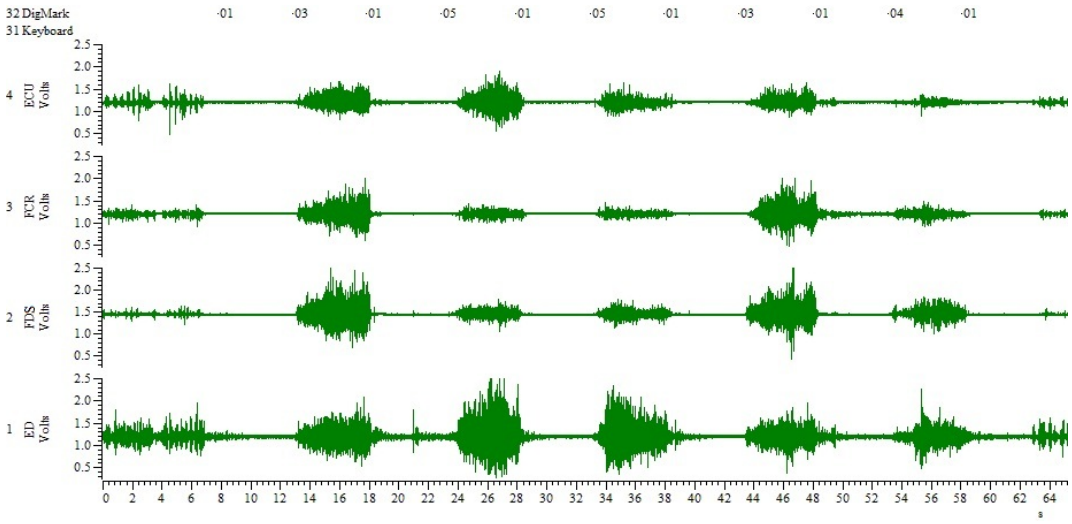

Figure 7.2: Trial 13 from female test subject. Note inconsistencies in EMG over duration of gesture. Activity prior to first digital marker or after the last marker was random flexions and not considered during signal processing.

This trial shows a grip, two tripods, another grip, and lastly a pinch. As can be seen the subject does not always immediately form the gesture. The signal is also not always consistent throughout the gesture. In addition, the pinch gesture showed some lack of activity.

7.2 Fourier Transforms

The following are the Fourier transforms for each gesture of one subject. These ones were chosen to show since they show the features of each sensor channel well, and are rather clear.

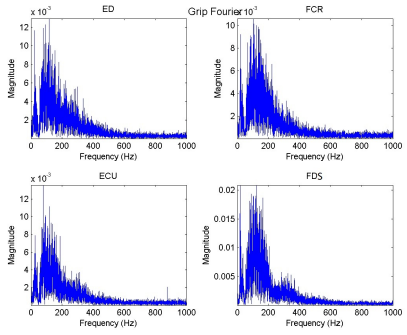

Figure 7.3: Grip Fourier transforms

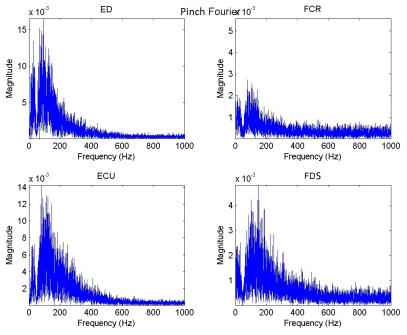

Figure 7.4: Pinch Fourier transforms

With all of the sensor channels the main power of the signal was located around the 50 Hz to 150 Hz range. This was the case across all gestures and all subjects. With the grip gesture, we can see that the ED, ECU, and FCR sensors all had roughly the same magnitudes in their frequency powers of around 0.01 - 0.012. The FDS was somewhat higher

on this subject at 0.02, but with other subjects it was closer to even across all channels for grip.

With the pinch gesture, the extensors showed greater maximum power than the flexor muscles. This is because the main power in this gesture is coming from the three fingers being extended when performing it, so we should expect to see greater activity there. The ED had a maximum magnitude of about 0.015 and the ECU had a maximum of about 0.014. The FCR had a maximum of about 0.003, and the FDS at about 0.005.

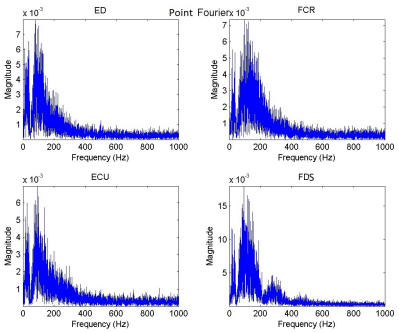

Figure 7.5: Point Fourier transforms

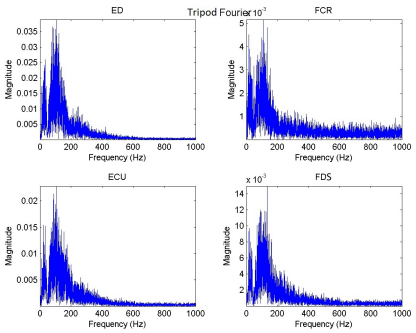

Figure 7.6: Tripod Fourier transforms

Point, like grip, usually has similar maximum magnitudes across all muscles. However, it is usually a much lower intensity than the grip gesture and in fact normally provided the weakest signals out of all gestures. Here, we can see the maximum magnitudes was around 0.006 - 0.007, with FDS being the largest at around 0.015.

Lastly, we have tripod. Like with the pinch gesture, we see that the extensors provide the largest maximum magnitude. The ED provided a maximum of 0.035 and the ECU a maximum of around 0.02. The FCR maximum was 0.004 and the FDS was 0.015. Tripod was generally rather similar to pinch in most subjects when looking at just the time-domain signal. However, we can see when performing the Fourier transform that the tripod gesture generally provided a slightly higher maximum power.

7.3 Power Spectral Densities

These are the power spectral densities from the same subject and the same gestures. We can see a peak of -40dB of power for three channels on the grip gesture. The FDS was slightly higher, but overall the maximum power was fairly even across all channels for this gesture.

For the pinch gesture we had about -40dB for the extensor muscles, but lower on the FCR at just over -60dB and -50dB for the FDS. The point PSD sgowed a quite similar pot to that of the grip gesture. The FCR and ECU was slightly lower than the grip gesture at about -50dB, but the ED and FDS matched up with the grip gesture.

The tripod gesture showed the ED peaking at about -30dB and the ECU at a little more than -40dB. The FCR was less than -50dB and the FDS peaked right at -40dB.

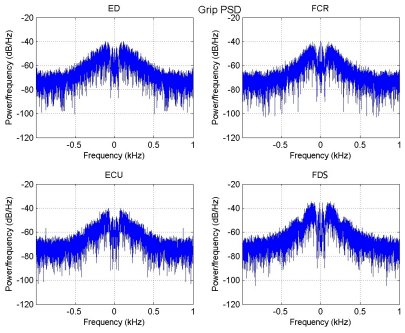

Figure 7.7: Grip PSD

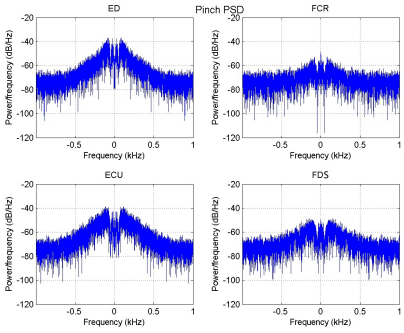

Figure 7.8: Pinch PSD

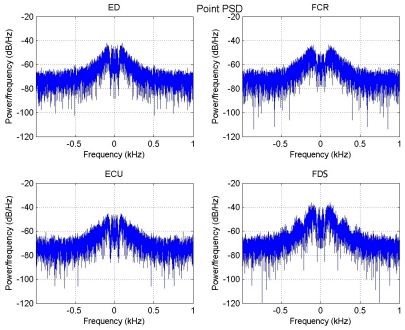

Figure 7.9: Point PSD

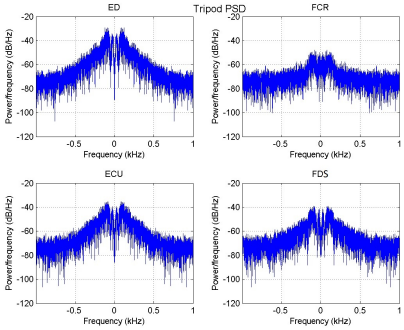

Figure 7.10: Tripod PSD

7.4 Spectrograms

The spectrograms here help to bring out and emphasize the power in relation to the frequency components. Like with the previous transforms, we see that except for the FDS being slightly more powerful the signals are roughly even.

With pinch, we again see that the majority of the gestures' power lies with the extensor muscles. The FDS showed some interesting pattern with some higher frequency components getting more power, but this did not seem consistent across the subjects.

The point spectrogram looks similar to that of the grip, just with less maximum powers as expected. The FDS looks quite similar though except for a band of lower power frequency components around the 250 Hz area.

The tripod too looks similar to the pinch spectrogram, but with greater power components.

7.5 Signal RMS Feature Distribution

The RMS values for each data channel on each gesture was calculated and then plotted against each other. This figure shows the grouping of these values as a scatter plot. The plot is a two-dimensional figure, but since we have four channels of data we have to split up the plot into sixteen different plots. Each plot has a different set of axes, to show the variations

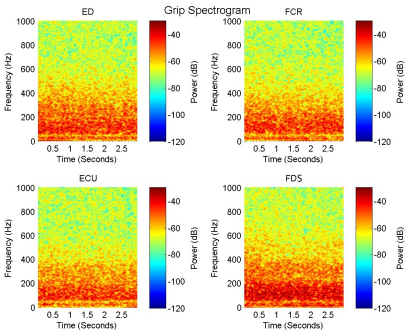

Figure 7.11: Grip PSD

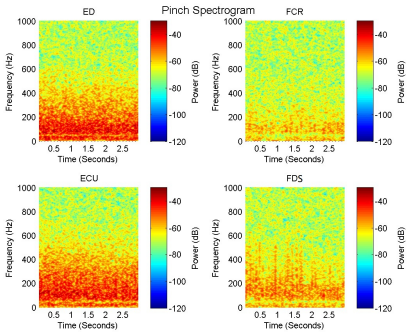

Figure 7.12: Pinch PSD

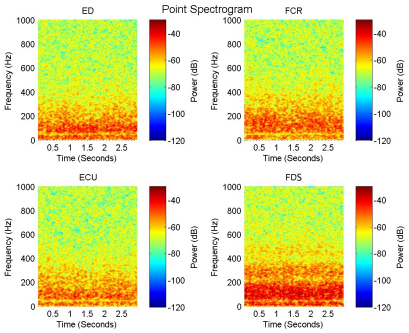

Figure 7.13: Point PSD

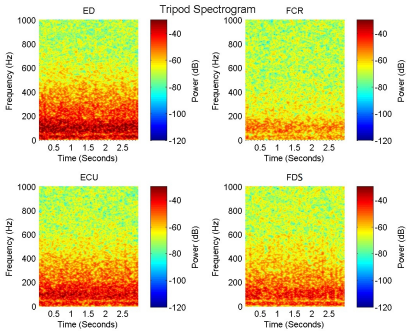

Figure 7.14: Tripod PSD

for each feature against the other.

In figure 7.15, the pinch gesture is represented by blue circles, the grip gestures by red squares, point is represented by green triangles and tripod is yellow diamonds. Each of the gestures show some grouping, which is something that is desired for pattern recognition techniques to work.

7.6 Classifier Scoring

We tried scoring the classifier with two methods. The first method we used was a confusion matrix. This shows the true class labels versus the guessed class labels from the classifier. Figure 7.16 shows our results from using a KNN classifier trained with one set of data and then tested using another set from the same subject. The second scoring method we used was a ROC curve. Figure 7.17 shows the ROC

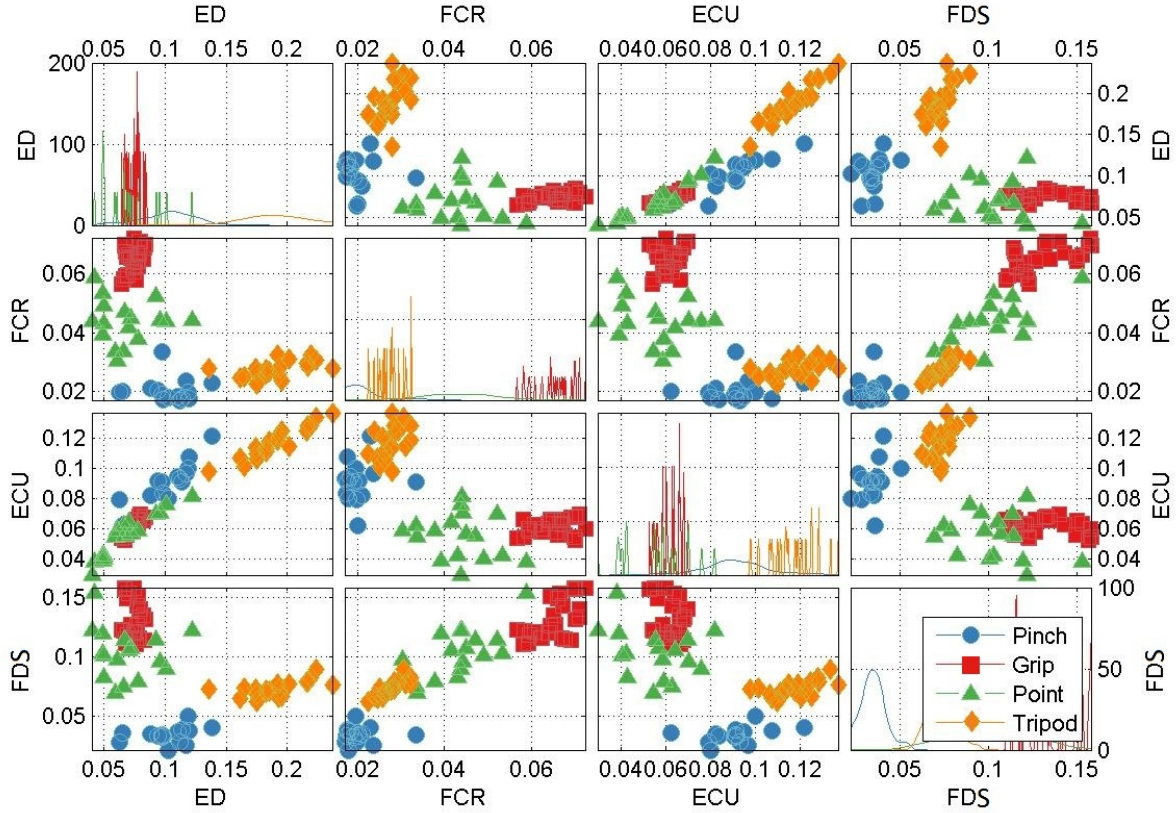

Figure 7.15: RMS values feature distribution. The features are shown in several 2D plots. This helps to show the clustering of different gestures by each feature. We can see that while grip is very tightly clustered, point tends to be more scattered. Having more features however allows us to look at each gesture from multiple perspectives, thus increasing the possibility of correctly identifying more problematic gestures.

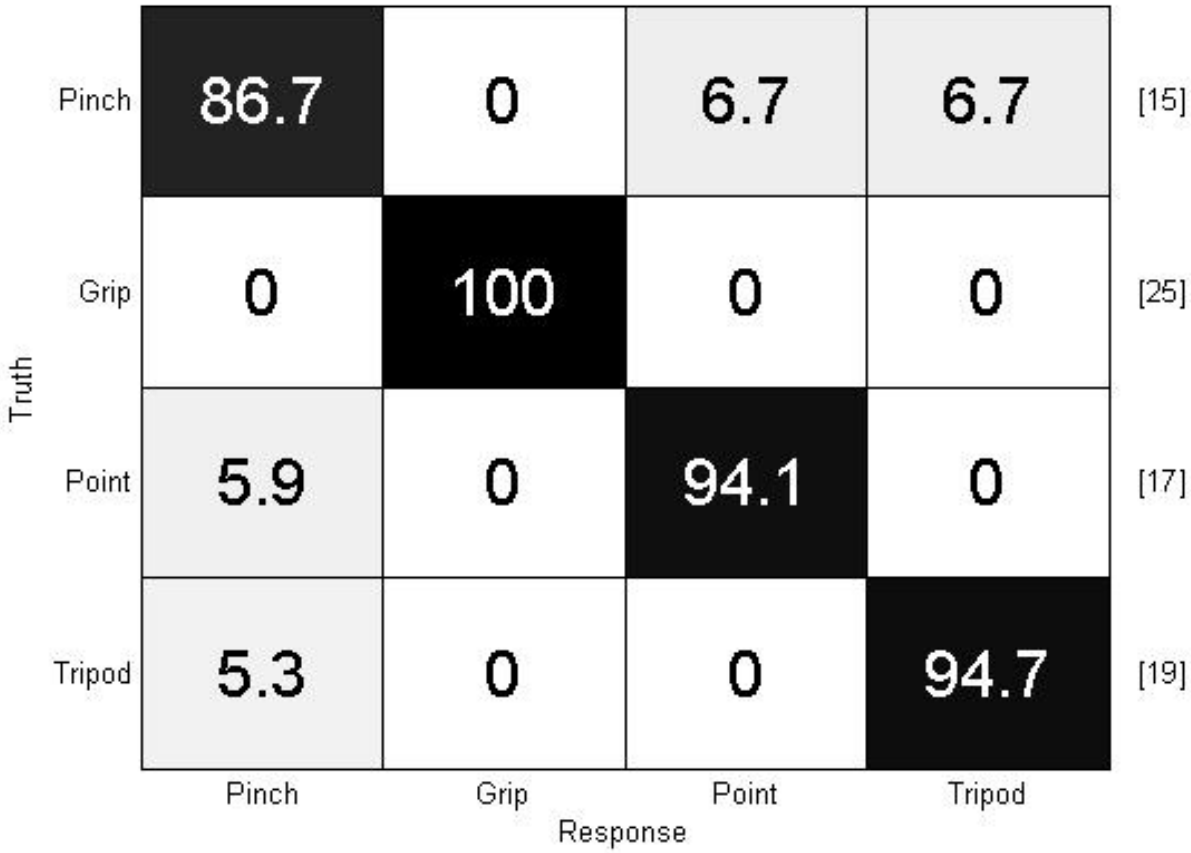

Figure 7.16: 4x4 confusion matrix showing the identifications of each gesture. The vertical axis shows the true values of the inputs with the number of each input displayed on the right hand side. The horizontal axis shows the output response of the classifier. The plot should be read horizontally. For example, the classifier identified 86.7% of pinch gestures correctly, but misidentified 6.7% as point and another 6.7% as tripod.

8 Discussion

8.1 Use of Four Sensors

The current iteration of the Touch Bionics iLIMB utilises two surface EMG sensors in order to control the prosthesis. While this works fine for the current method of control by reading a series of wrist muscle felxions and extensions, implementing a pattern recognition method of control with only these two sensors has been shown to be inadequate. The literature shows that with only two sensors there is not enough data for a DSP to properly identify different gestures and that classification errors would be common. The same article goes on to state that utilising any less than three sensors would lead to an unacceptable amount of classification errors and that the optimal number would be three to five sensors. However, they stated that any more would result in diminishing returns and would likely not prove to be cost effective. The importance of using four sensors over two is evident when attempting to classify certain gestures. When using only two sensors, it could be sometimes difficult when differentiating gestures as their EMG data looked similar enough that they could be confused for each other. The addition of the two additional sensors however doubled the amount of unique EMG data we were recording for each gesture. The addition of EMG data from these sensors allowed for the gestures to be more uniquely defined, and so more easily identified based on their EMG signature. While we appreciate the need to keep the number of sensors being employed to a minimum in order to keep the cost of the prosthesis down and to reduce the processing power needed, it seems quite unlikely that use of only two sensors could provide the necessary amount of information to accurately control the prosthesis.

Now that it had been determined that we would be using four sensors to collect data, the decision had to be made on their placements on the forearm. The locations used when using only two sensors, the flexor carpi radials and extensor digitorum, seemed to provide useful data already so it did not seem to be any reason not to continue using these two locations. However, for the additional two sensors we needed to see what may work best. Initially, we tried placing two sensors on each muscle with one sensor orthogonal to the other. This proved to be inadequate and difficult to work with. The main difficulty was that the sensors would have to be in very close proximity to each other for this, and so crosstalk between them was a concern. Mogk et al. [23] showed that there was approximately 40% common signal found between sensors within 3 cm distance of each other. We would like to reduce cross talk since our goal is to find more unique signal profiles for each sensor. Additionally, the width of the muscles themselves, especially with the FCR, may have been thinner than the sensor. This would lead to the sensor placed orthogonally to not detect activity in the muscle during contraction, or to noise coming off of other muscles. Because of these limiting factors, it seemed that it would be much more useful to use four separate muscles of the forearm as locations for the sensors.

The two additional muscles chosen to be sensor sites had to fulfill certain criteria that would accommodate as many forearm amputees as possible. One concern with these muscles was their proximity to the elbow joint. We determined that muscles nearest the elbow joint

would be most desirable since a wider range of people with different residual limb lengths could still control the prosthesis. Additionally, it was desirable to focus on muscles that had some influence on digit extension or flexion. Lastly, it was thought that the number of flexor and extensor muscles being monitored should be kept even. Due to the nature of surface EMG sensors, it should also be noted that muscles that are superficial and near the surface of the skin are the only viable muscles that can be used.

The two additional muscles that were decided to be used were the extensor carpi ulnaris and the flexor digitorum superficialis. Both of these muscles have some influence over moving the digits and both have their main masses proximal to the elbow joint. The extensor carpi ulnaris (ECU) is also superficial to other muscles in the forearm which makes it ideal for surface EMG readings. The number of fingers the flexor digitorum superficialis (FDS) manipulates makes it a good muscle for monitoring finger flexion. Both of these muscles aid in the movement of various digits of the hand which makes them useful candidates for EMG monitoring of them for control of the prosthesis. The FDS influences hand gestures by flexing the distal interphalangeal joints of the four fingers [25], provides a good addition of data for when the fingers are being flexed, such as with the grip or point gestures. While the ECU is primarily used for extending the wrist with ulnar deviation, we also observed that it tended to extend the fifth metacarpal, which would be useful in identifying the pinch gesture.

8.1.1 Fourier Transform

What we can definitely conclude from the Fourier transforms is that the main frequency components of any of these EMG signals lies in the 20Hz to 150Hz range or so, with slight variations depending on the gesture being performed and the muscle being monitored. As expected, we also see the effects of the 50Hz notch filter where there is a large gap in the plot around that frequency range across all of our recordings.

The best features we could extract from here would likely be the maximum magnitudes of each channel. These seemed to differ enough that each gesture had a unique combination of which muscle was providing the greatest power to the gesture and by how much. This seems to be a quite promising transform for feature extraction.

8.1.2 Power Spectral Density

The PSD plots tended to emphasize the actual peak power of the signals, which could be useful features to use. However, we can also easily calculate the average and RMS power values of the signals. The RMS values in particular are often used as features in pattern recognition of signals. If the DSP being used will be calculating the RMS values, it may be unnecessary to be calculating other transforms.

8.1.3 Spectrogram

The spectrogram more easily shows the power distribution of the signal power over the different frequency components, so we can see the differences a little easier between the different gestures. It seems to more easily emphasize the power distribution across

the frequency spectrum than just the PSD or Fourier transform, at least from a visual perspective. However, the calculation time for this seemed much greater which could put a not insignificant delay on the processing time of the prosthesis. It also seems like that a spectrogram would provide no better feature extraction possibilities, so it would likely be just a waste of processing power to utilise it.

8.2 The Confusion Matrix

The confusion matrix we obtained in figure 7.16 showed a very good outcome for the data set. Pinch in fact showed the worst results, with 5.9% of the test data being identified as Point and 5.3% as Tripod. Other gestures were also misidentified as Pinch, notably 6.7% of Points and Tripod. When looking at the feature distribution plot in figure 7.15 we see a lot of spread in the Point gesture. This may be a contributing factor to the classification error. We also see some overlap between Tripod and Pinch in the FCR versus ECU and ED versus ECU plots. Grip also shows some considerable overlap with Point. However, the grouping might be tight enough that the classifier could easily enough discriminate between the two which would lead to a better score for Grip in the confusion matrix.

8.3 Analysis in the Frequency Domain

One important aspect to consider is whether or not it is actually beneficial to be trying to identify the gestures in the frequency domain rather than the time domain. The benefit of staying within the time domain is that we do not require any computational resources to be spent transforming the signal into the frequency domain, which may end up keeping monetary cost down as well. However, there are some drawbacks as well to staying within the time domain where working within the frequency domain wouldn't be a concern. The main drawback from extracting features from the time domain is that there is an added delay in the DSP when it is attempting to identify gesture formation during the onset of muscle contraction. We can see from the spectrograms that the power spectral density remains fairly constant as the subject holds the gesture, so as soon as a muscle contraction is detected we should be able to look at PSD to determine the gesture class. With the muscle contracture detection algorithm studied by Lee et al. [16], detection time should be brought to a minimum.

Barreto et al. [1] have also noted that sEMG signals from a muscle contraction can have a very different frequency composition than that due to a different muscle. They reference a study done for the U.S. Department of Health and Human Services which makes the claim that this is due to the differing contraction lengths for each muscle. [24] What this means is that we can see more information about the signal in the frequency domain than what we would likely be able to in just the time domain.

8.4 Pattern Recognition

The next step to take with this research would to look in to pattern recognition techniques and apply them to the gathered data to test the feasibility of using these techniques. What we did here is mainly a cursory look at what could be done in this area. The feature distribution from the RMS values of the signals seem to be well grouped for each gesture. The point gesture seems to have some spread in a few of the feature comparison plots but still seems rather distinct. The other three gestures seem to be rather tightly grouped, which indicates that these data would likely be good for use in pattern recognition.

For the moment, while using a KNN classifier it seems that using these four data channels could be a promising method of controlling the prosthesis. What further needs to be done is to see if any other classifiers can perform better than KNN and if making use of preprocessing such as PCA would improve results. ANOVA was shown by Shuman [30] to be a reliable method of reducing features. Further testing with different classifiers may also show that other features than what we used here would work better. Another thing to consider would be the addition of more gestures than what we tested for. There are several other gestures that some forearm prostheses can make use of that would be good to look in to as well.

8.5 Future Prospects of Myolelectric Prostheses

While this research focused mainly on the signal processing side of myoelectric prosthesis control, there are advances being made on the circuitry and hardware aspects of these prosthetics. Merrill et al. [22] have done some research into utilising permanent EMG sensors that are implanted under the surface of the skin and directly onto the muscle. They theorise that having sensors like this would be preferable over surface sensors since data from surface sensors often contain activity from multiple muscle sources. The implantable sensors they propose would be capable of sensing a single principal component. The size of these sensors would also be much smaller, and as such prosthetists would be able to make use of more. Better sensing of principal components could possibly make muscle features much more unique.

For upper limb amputees that have little to no residual musculature, current myoelectric prostheses would be of no help. There have been techniques developed though, which promise to help these amputees. Targeted muscle reinvigoration is a process whereby the certain skeletal muscles have their nerve connexions severed. [11] The surgeon then attaches the nerves that would normally go to the amputated muscles and connect these to the prior muscles that were disconnected. This allows the amputee to flex those muscles as if they were flexing their amputated limbs, allowing for EMG signals to be acquired.

Another alternative sensor was studied by Rossini et al. [28] The sensor they studied in fact did not make use of EMG signals at all. Instead, their sensor took the signal directly from the nervous system to utilise as a control signal. This would be very beneficial for those amputees who do not have residual limb muscles which could provide a useful EMG signal. It also holds the possibility of sensory feedback, opening up the possibility of prostheses that are even more natural. Their study however showed degredation of the nerves after ten days. This would be important to overcome if these sensors were to be used in future applications.

9 Summary and Conclusion

The data seems to be quite promising with regards to the feature extraction of these gestures. More research needs to be done in order to determine the optimal pattern recognition techniques, but with good features being extracted this should not be a problem. We feel that for the moment, using the RMS values of each signal as our main feature for each channel will provide the fastest method of calculation and will work well with a DSP.

It may be determined that other muscles or sensor locations will provide better data, but our current set up seemed to provide all of the data we would need at a level of quality needed. It is unknown what kind of effect filtering and rectification of the signal would have on these results, which is something that would need to be taken into account and tested for before commercial application. Overall though, we gathered enough useful data with the tools and techniques available to us.

Future work that can be done specifically with regards to this study would be to look into the benefits of using other transforms for feature extraction. We have already mentioned DWT and MWM which have shown some promise, but there are a large number of others that could be done as well. What really needs to be done now is the building and evaluation of different classifiers with different feature sets to determine the most optimal ones. Further on, we would also want to look at developing better realtime analysis tools to speed up detection times and reduce lag between the amputee signaling to perform the gesture and the prosthesis actually performing it.

This research experiment was undertaken to gather evidence that it would be possible to extract useful features from sEMG signals of the forearm that can be used in the control of a myoelectric prosthesis. Much of the experiment had to take into account differing levels of forearm amputation and the capabilities of both the subjects and the current technologies being employed. The features being extracted had to be unique enough the a signal processor in the prosthesis will be able to quickly and accurately identify the gesture being performed.

Appendices

MATLAB Scripts

Epoching Script

```
function [ ] = epoch(ED_signal, FCR_signal, ECU_signal, FDS_signal, ...
    DigMark, base_event)
%EPOCHSMR:  Epochs each gesture in the given signal, and then
%           corrects for the rest signal offset
%   Ex.- ED_signal: e1_ED
%        FCR_signal: e1_FCR
%        DigMark: e1_DigMark
%   2000: Sampling frequency

%% Epoching each gesture
event_time1 = DigMark.times(2,1);
event_offset1 = round(event_time1 * 2000);
start1 = 3000 + event_offset1;
stop1 = start1 + 6000;

epoch1 = [ED_signal.values(start1:stop1) ...
    FCR_signal.values(start1:stop1) ECU_signal.values(start1:stop1) ...
    FDS_signal.values(start1:stop1)];
epoch1(:,1) = epoch1(:,1);
epoch1(:,2) = epoch1(:,2);
epoch1(:,3) = epoch1(:,3);
epoch1(:,4) = epoch1(:,4);

event_time2 = DigMark.times(4,1);
event_offset2 = round(event_time2 * 2000);
start2 = 3000 + event_offset2;
stop2 = start2 + 6000;

epoch2 = [ED_signal.values(start2:stop2) ...
    FCR_signal.values(start2:stop2) ECU_signal.values(start2:stop2) ...
    FDS_signal.values(start2:stop2)];
epoch2(:,1) = epoch2(:,1);
epoch2(:,2) = epoch2(:,2);
epoch2(:,3) = epoch2(:,3);
epoch2(:,4) = epoch2(:,4);

event_time3 = DigMark.times(6,1);
event_offset3 = round(event_time3 * 2000);
start3 = 3000 + event_offset3;
stop3 = start3 + 6000;

epoch3 = [ED_signal.values(start3:stop3) ...
    FCR_signal.values(start3:stop3) ECU_signal.values(start3:stop3) ...
    FDS_signal.values(start3:stop3)];
```

```
epoch3(:,1) = epoch3(:,1);
epoch3(:,2) = epoch3(:,2);
epoch3(:,3) = epoch3(:,3);
epoch3(:,4) = epoch3(:,4);

event_time4 = DigMark.times(8,1);
event_offset4 = round(event_time4 * 2000);
start4 = 3000 + event_offset4;
stop4 = start4 + 6000;

epoch4 = [ED_signal.values(start4:stop4) ...
    FCR_signal.values(start4:stop4) ECU_signal.values(start4:stop4) ...
    FDS_signal.values(start4:stop4)];
epoch4(:,1) = epoch4(:,1);
epoch4(:,2) = epoch4(:,2);
epoch4(:,3) = epoch4(:,3);
epoch4(:,4) = epoch4(:,4);

event_time5 = DigMark.times(10,1);
event_offset5 = round(event_time5 * 2000);
start5 = 3000 + event_offset5;
stop5 = start5 + 6000;

epoch5 = [ED_signal.values(start5:stop5) ...
    FCR_signal.values(start5:stop5) ECU_signal.values(start5:stop5) ...
    FDS_signal.values(start5:stop5)];
epoch5(:,1) = epoch5(:,1);
epoch5(:,2) = epoch5(:,2);
epoch5(:,3) = epoch5(:,3);
epoch5(:,4) = epoch5(:,4);

% Setting the rest signal
rest_time = DigMark.times(base_event,1);
rest_offset = round(rest_time * 2000);
rest_stop = rest_offset - 1;
rest_start = rest_stop - 6000;

rest_signal = [ED_signal.values(rest_start:rest_stop) ...
    FCR_signal.values(rest_start:rest_stop) ...
    ECU_signal.values(rest_start:rest_stop) ...
    FDS_signal.values(rest_start:rest_stop)];
rest_signal(:,1) = rest_signal(:,1);
rest_signal(:,2) = rest_signal(:,2);
rest_signal(:,3) = rest_signal(:,3);
rest_signal(:,4) = rest_signal(:,4);

%Correcting each epoch for rest offset
new_ED1 = epoch1(1:6001,1) - rest_signal(1:6001,1);
new_FCR1 = epoch1(1:6001,2) - rest_signal(1:6001,2);
new_ECU1 = epoch1(1:6001,3) - rest_signal(1:6001,3);
new_FDS1 = epoch1(1:6001,4) - rest_signal(1:6001,4);
correctedSignal1 = [new_ED1 new_FCR1 new_ECU1 new_FDS1];

new_ED2 = epoch2(1:6001,1) - rest_signal(1:6001,1);
```

```
new_FCR2 = epoch2(1:6001,2) - rest_signal(1:6001,2);
new_ECU2 = epoch2(1:6001,3) - rest_signal(1:6001,3);
new_FDS2 = epoch2(1:6001,4) - rest_signal(1:6001,4);
correctedSignal2 = [new_ED2 new_FCR2 new_ECU2 new_FDS2];

new_ED3 = epoch3(1:6001,1) - rest_signal(1:6001,1);
new_FCR3 = epoch3(1:6001,2) - rest_signal(1:6001,2);
new_ECU3 = epoch3(1:6001,3) - rest_signal(1:6001,3);
new_FDS3 = epoch3(1:6001,4) - rest_signal(1:6001,4);
correctedSignal3 = [new_ED3 new_FCR3 new_ECU3 new_FDS3];

new_ED4 = epoch4(1:6001,1) - rest_signal(1:6001,1);
new_FCR4 = epoch4(1:6001,2) - rest_signal(1:6001,2);
new_ECU4 = epoch4(1:6001,3) - rest_signal(1:6001,3);
new_FDS4 = epoch4(1:6001,4) - rest_signal(1:6001,4);
correctedSignal4 = [new_ED4 new_FCR4 new_ECU4 new_FDS4];

new_ED5 = epoch5(1:6001,1) - rest_signal(1:6001,1);
new_FCR5 = epoch5(1:6001,2) - rest_signal(1:6001,2);
new_ECU5 = epoch5(1:6001,3) - rest_signal(1:6001,3);
new_FDS5 = epoch5(1:6001,4) - rest_signal(1:6001,4);
correctedSignal5 = [new_ED5 new_FCR5 new_ECU5 new_FDS5];


uisave({'epoch1','epoch2','epoch3','epoch4','epoch5','rest_signal',
'correctedSignal1','correctedSignal2','correctedSignal3','correctedSignal4',
'correctedSignal5'},'e');
end
```

Fourier Transform Script

```
function [ signal_fft ] = freqSpec_SMR( dataFile, max, gesture_type, ...
    file_path, filename)

%FREQSPEC Summary of this function goes here
%   Detailed explanation goes here

N = 2^nextpow2(6000);
signal_fft = fft(dataFile, N)/6000;

f = 2000/2 * linspace(0, 1, N/2 +1);

subplot(2,2,1);plot(f,2*abs(signal_fft(1:N/2+1,1)));
ylim([0 .03]);
axis tight;
view(0,90);
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('ED');

subplot(2,2,2);plot(f,2*abs(signal_fft(1:N/2+1,2)));
ylim([0 .03]);
```

```
axis tight;
view(0,90);
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('FCR');

subplot(2,2,3);plot(f,2*abs(signal_fft(1:N/2+1,3)));
ylim([0 .03]);
axis tight;
view(0,90);
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('ECU');

subplot(2,2,4);plot(f,2*abs(signal_fft(1:N/2+1,4)));
ylim([0 .03]);
axis tight;
view(0,90);
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('FDP');
%%
if max==1
    switch(gesture_type)
        case 2
            mtit('Pinch Fourier, MAX','fontsize',13);
        case 3
            mtit('Grip Fourier, MAX','fontsize',13);
        case 4
            mtit('Point Fourier, MAX','fontsize',13);
        case 5
            mtit('Tripod Fourier, MAX','fontsize',13);
    end
else
                switch(gesture_type)
        case 2
            mtit('Pinch Fourier','fontsize',13);
        case 3
            mtit('Grip Fourier','fontsize',13);
        case 4
            mtit('Point Fourier','fontsize',13);
        case 5
            mtit('Tripod Fourier','fontsize',13);
                end
end
saveas(gcf,fullfile(file_path,filename),'tiff');

end
```

Periodogram and Average Power Script

```
function [ ] = avgpow(signal,max,gesture_type,file_path,filename)
%avgpow: Computes the periodogram for each channel along with average power
%for each channel, and saves the plot and values to the drive
```

```
4
5  % Create periodogram
6  hp = spectrum.periodogram('hamming');
7  Fs = 2000;
8
9  ED =signal(:,1);
10 FCR = signal(:,2);
11 ECU = signal(:,3);
12 FDS = signal(:,4);
13
14 % Create options object and set properties
15 hpopts_ED = psdopts(hp,ED);
16 set(hpopts_ED,'Fs',Fs,'SpectrumType','twosided','centerdc',true);
17
18 hpopts_FCR = psdopts(hp,FCR);
19 set(hpopts_FCR,'Fs',Fs,'SpectrumType','twosided','centerdc',true);
20
21 hpopts_ECU = psdopts(hp,ECU);
22 set(hpopts_ECU,'Fs',Fs,'SpectrumType','twosided','centerdc',true);
23
24 hpopts_FDS = psdopts(hp,FDS);
25 set(hpopts_FDS,'Fs',Fs,'SpectrumType','twosided','centerdc',true);
26
27 %Compute the power spectral densities for each channel
28 hpsd_ED = psd(hp,ED,hpopts_ED);
29 hpsd_FCR = psd(hp,FCR,hpopts_FCR);
30 hpsd_ECU = psd(hp,ECU,hpopts_ECU);
31 hpsd_FDS = psd(hp,FDS,hpopts_FDS);
32
33 %Plots PSD
34 subplot(2,2,1); plot(hpsd_ED); set(gca,'YLim',[-120 -20]);title('ED');
35 subplot(2,2,2); plot(hpsd_FCR);set(gca,'YLim',[-120 -20]);title('FCR');
36 subplot(2,2,3); plot(hpsd_ECU);set(gca,'YLim',[-120 -20]);title('ECU');
37 subplot(2,2,4); plot(hpsd_FDS);set(gca,'YLim',[-120 -20]);title('FDS');
38 if max==1
39     switch(gesture_type)
40         case 2
41             mtit('Pinch PSD, MAX','fontsize',13);
42         case 3
43             mtit('Grip PSD, MAX','fontsize',13);
44         case 4
45             mtit('Point PSD, MAX','fontsize',13);
46         case 5
47             mtit('Tripod PSD, MAX','fontsize',13);
48     end
49 else
50                 switch(gesture_type)
51         case 2
52             mtit('Pinch PSD','fontsize',13);
53         case 3
54             mtit('Grip PSD','fontsize',13);
55         case 4
56             mtit('Point PSD','fontsize',13);
57         case 5
```

```
            mtit('Tripod PSD','fontsize',13);
                end
end
saveas(gcf,fullfile(file_path,filename),'tiff');

%Obtains average power for each data channel
power_freqdomain_ED = avgpower(hpsd_ED);
power_freqdomain_FCR = avgpower(hpsd_FCR);
power_freqdomain_ECU = avgpower(hpsd_ECU);
power_freqdomain_FDS = avgpower(hpsd_FDS);
%Converting each average power to dB
power_freqdomain = [power_freqdomain_ED power_freqdomain_FCR ...
    power_freqdomain_ECU power_freqdomain_FDS];
avg_power = 10*log10(power_freqdomain/2);

uisave({'avg_power'},'')
end
```

Spectrogram Script

```
function [ ] = spectr(file, gesture_type, max, filename, file_path)
%SPECTR Summary of this function goes here
%   file: gesture being processed
%   gesture_type: gesture # as indicated
%   max: if gesture is maximum contraction, values should be 1. Otherwise,
%   anything else
%   filename: desired filename
%   file_path: path to folder where jpg will be saved

nfft = 2^nextpow2(6000); %Length of the FFT

%%
%Calculating the soectrograms for each channel
[S1,F1,T1,P1] = spectrogram(file(:,1),256,250,nfft,2000);
[S2,F2,T2,P2] = spectrogram(file(:,2),256,250,nfft,2000);
[S3,F3,T3,P3] = spectrogram(file(:,3),256,250,nfft,2000);
[S4,F4,T4,P4] = spectrogram(file(:,4),256,250,nfft,2000);

%%
%Plotting spectrograms
subplot(2,2,1); surf(T1,F1,10*log10(P1),'EdgeColor','none');
c1 = colorbar; caxis([-120 -30]); ylabel(c1,'Power (dB)');
axis tight;
view(0,90);
xlabel('Time (Seconds)'); ylabel('Frequency (Hz)');
title('ED');

subplot(2,2,2); surf(T2,F2,10*log10(P2),'EdgeColor','none');
c2 = colorbar; caxis([-120 -30]); ylabel(c2,'Power (dB)');
axis tight;
view(0,90);
```

```
xlabel('Time (Seconds)'); ylabel('Frequency (Hz)');
title('FCR');

subplot(2,2,3); surf(T3,F3,10*log10(P3),'EdgeColor','none');
c3 = colorbar; caxis([-120 -30]); ylabel(c3,'Power (dB)');
axis tight;
view(0,90);
xlabel('Time (Seconds)'); ylabel('Frequency (Hz)');
title('ECU');

subplot(2,2,4); surf(T4,F4,10*log10(P4),'EdgeColor','none');
c4 = colorbar; caxis([-120 -30]); ylabel(c4,'Power (dB)');
axis tight;
view(0,90);
xlabel('Time (Seconds)'); ylabel('Frequency (Hz)');
title('FDS');


%%
if max==1
    switch(gesture_type)
        case 2
            mtit('Pinch Spectrogram, MAX','fontsize',13);
        case 3
            mtit('Grip Spectrogram, MAX','fontsize',13);
        case 4
            mtit('Point Spectrogram, MAX','fontsize',13);
        case 5
            mtit('Tripod Spectrogram, MAX','fontsize',13);
    end
else
                switch(gesture_type)
        case 2
            mtit('Pinch Spectrogram','fontsize',13);
        case 3
            mtit('Grip Spectrogram','fontsize',13);
        case 4
            mtit('Point Spectrogram','fontsize',13);
        case 5
            mtit('Tripod Spectrogram','fontsize',13);
                end
end
saveas(gcf,fullfile(file_path,filename),'tiff');
end
```

Bibliography

[1] A.B. Barreto; S.D.; P.A. Scargle; M. Adjouadi. A practical emg-based human-computer interface for users with motor disabilities. *Journal of Rehabilitation Research and Development*, 37, No.1:53–64, 2000.

[2] Touch Bionics. i-limb ultra data sheet. `http://www.touchbionics.com/media/2210/i-limb%20ultra%20data%20sheet%20lo-res.pdf`.

[3] Touch Bionics. i-limb ultra order form. `http://www.cascade-usa.com/customer/caorsu/customerpages/pdf/custom_order_forms/i-limb_ultra.pdf`.

[4] Touch Bionics. i-limb ultra service and fitting manual. `http://www.touchbionics.com/media/48359/i-limb_ultra_service_and_fitting_manual.pdf`.

[5] Touch Bionics. Touch bionics sensors spec (low res). Sensor specifications and circuit diagram.

[6] Reza Boostani and Mohammad Hassan Moradi. Evaluation of the forearm emg signal features for the control of a prosthetic hand. *Physiological Measurement*, 24:309–319, 2003.

[7] H. Bouwsema, C.K. van der Sluis, and R.M. Bongers. Learning to control opening and closing a myoelectric hand. *Archives of physical medicine and rehabilitation*, 91.9:1442–1446, 2010.

[8] Mark Broomfield. Congenital limb deficiency. The UK Limb Loss Information Centre, http://limblossinformationcentre.com/parent-centre/coming-to-terms-with-congenital-limb-loss/congenital-limb-deficiency/.

[9] TR Dillingham, LE Pezzin, and EJ MacKenzie. Limb amputation and limb deficiency: epidemiology and recent trends in the united states. *Southern Medical Journal*, 95(8):875–83, 2002 Aug.

[10] Jun-Uk Chu et el. Myoelectric hand prosthesis with novel adaptive grasping and self-locking. *INTERNATIONAL JOURNAL OF PRECISION ENGINEERING AND MANUFACTURING*, 12, No. 6:1095–1103, 2011 Dec.

[11] Todd A. Kuiken et el. Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms. *JAMA*, 301, No. 6:619–628, 2009 Feb.

[12] Delsys Gianluca De Luca. Fundamental concepts in emg signal acquisition. 2001, Rev. 2003. `http://www.delsys.com/Attachments_pdf/WP_Sampling1-4.pdf`.

[13] G. Hefftner, W. Zucchini, and G. Jaros. The electromyogram (emg) as a control signal for functional neuromuscular stimulation. i. autoregressive modeling as a means of emg signature discrimination. *IEEE Transactions on Biomedical Engineering*, 35.4:230–237, 1988.

[14] Z. Khokhar, Z. Xiao, and C. Menon. Surface emg pattern recognition for real-time control of a wrist exoskeleton. *BioMedical Engineering OnLine*, 9.1, 2010.

[15] J. Kilby and H.G. Hosseini. Wavelet analysis of surface electromyography signals. *Proceedings of the 26th Annual International Conference of the IEEE EMBS San Francisco, CA, USA*, Sept 1-5, 2004.

[16] Angela S. Lee, Jacek Cholewicki, and N. Peter Reeves. The effect of background muscle activity on computerized detection of semg onset and offset. *Journal of Biomechanics*, 40:3521–3526, 2007.

[17] A. Leis and V. Trapani. *Atlas of Electromyography.* Oxford, Oxford UP, 2000.

[18] G. Li and A. Kuiken. Emg pattern recognition control of multifunctional prostheses by transradial amputees. *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6914–6917, 2009.

[19] G. et al. Li. Conditioning and sampling issues of emg signals in motion recognition of multifunctional myoelectric prostheses. *Annals of Biomedical Engineering*, 39.6:1779–1787, 2011.

[20] F. Martini, J. Nath, and E. Bartholomew. *Fundamentals of Anatomy and Physiology*, volume 9. Pearson Education, San Francisco, CA, USA, 2011. 289-297.

[21] R. Merletti and P. Parker. *Electromyography: Physiology, Engineering, and Noninvasive Applications.* IEEE, Piscataway, NJ, 2004.

[22] J; Troyk P.R.; Weir R.F.; Merrill, D.R.; Lockhart and D.L. Hankin. Development of an implantable myoelectric sensor for advanced prosthesis control. *Artificial Organs*, 35(3):249 – 252, 2011.

[23] J.P.M. Mogk and P.J. Keir. Crosstalk in surface electromyography of the proximal forearm during gripping tasks. *Journal of Electromyography and Kinesiology*, 13:63 – 71, 2003.

[24] B. LeVeau; G. Andersson; U.S. Department of Health and Human Services. Output forms: Data analysis and applications. *Selected Topics in Surface Electromyography for Use in the Occupational Setting: Expert Perspectives*, pages 70–102, 1992.

[25] Henry L.; Pease, William S.; Lew and Ernest W. Johnson. *Johnson's Practical Electromyography, Fourth edition.* Lippincott Williams & Wilkins, 2007.

[26] B.; et al. Peerdeman. Myoelectric forearm prostheses: State of the art from a user-centered perspective. *Journal of Rehabilitation Research and Development*, 48.6:719–738, 2011.

[27] Robi Polikar. Pattern recognition. *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[28] Paolo; et al. Rossini. Double nerve intraneural interface implant on a human amputee for robotic hand control. *Clinical Neurophysiology*, 121:777–783, 2010 Jan.

[29] J. Rafiee; M.A. Rafiee; F. Yavari; M.P. Schoen. Feature extraction of forearm emg signals for prosthetics. *Physiological Measurement*, 24:309–319, 2003.

[30] Gene Shuman. Using forearm electromyograms to classify hand gestures. *2009 IEEE International Conference on Bioinformatics and Biomedicine*, 19.2:261 – 264, 1-4 Nov. 2009.

[31] L. H; et al. Smith. Determining the optimal window length for pattern recognition-based myoelectric control: Balancing the competing effects of classification error and controller delay. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19.2:186–192, 2011.

[32] K. Englehart; B. Hudgins; P.A. Parker; M. Stevenson. Classication of the myoelectric signal using time-frequency based representations. *Medical Engineering & Physics*, 20:431–438, 1999.

[33] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, 3rd edition.* Elsevier Science & Technology, 2006.

[34] Peter Torrione, Sam Keene, and Kenneth Morton. PRT: The pattern recognition toolbox for MATLAB, 2011. Software available at `http://newfolderconsulting.com/prt`.

[35] Rutgers University. Lecture 15: Muscles of the appendicular skeleton i. `http://www.rci.rutgers.edu/~uzwiak/AnatPhys/APFallLect15.html`.