EMG Controlled Hand Prosthesis: EMG Classification System

by

Philip Chrapka

Electrical and Biomedical Engineering Design Project (4BI6)
Department of Electrical and Computer Engineering
McMaster University
Hamilton, Ontario, Canada

EMG Controlled Hand Prosthesis: EMG Classification System

by

Philip Chrapka

Electrical and Biomedical Engineering
Faculty Advisor: Prof. Sirouspour

Electrical and Biomedical Engineering Project Report
submitted in partial fulfillment of the degree of
Bachelor of Engineering

McMaster University
Hamilton, Ontario, Canada
April 23, 2010

Abstract

Since the 1970s, electromyographic control of a prosthetic device has been attempted in a number of different ways. It was only until recently, that the classification of electromyographic signals was possible through the use of neural networks. With the advent of more advanced techniques like support vector machines this type of control is becoming more realistic. This would initiate a great step in the development of prosthetic devices. It would become possible for more advanced control of these devices with a more natural interface. This project focuses on the development of an electromyographic classification system on an embedded platform for the specific use of controlling a prosthetic device.

Keywords - Electromyographic control, support vector machines, neural networks, custom hardware design

Acknowledgements

I would like to thank my project partners Phillip Kinsman and Shameem Bhatti for all their hard work and support while working on this project. I would also like to thank Prof. Sirouspour for his helpful guidance on certain aspects of this project. I cannot forget Prof. Niccolicci and his graduate students who were willing to provide our group with the use of their lab.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Definition

This project encompasses the design and implementation of an electromyograhic (EMG) signal classification system that is able to distinguish between different gestures. It is a portion of a group project whose overall goal is to be able to control a robotic hand prosthesis using EMG signals from the upper limb. The group consisted of Phillip Kinsman, Shameem Bhatti and myself.

1.2 Background

Disabilites due to amputations almost always impose severe loss of functionality in individuals and make their lives much more difficult. Of course, the amputated limb is replaced by a prosthetic device, nevertheless these have limited functionality and are often difficult to use. It would be very convenient to develop a practical prosthesis that can make use of the original EMG signals that were used to control the limb prior to the amputation to create a natural and seamless interface.

1.3 Objectives

The objective of this project was to design a system that can learn and discriminate between at least 3 sets of different hand gestures by analyzing the EMG signals elicited in the upper limb by these gestures. The chosen gestures were: opening and closing of the hand, flexion and extension of the wrist as well as pronation and supination. The result of the classification would be used as the input to control a robotic prosthesis. To ensure that this type of solution is indeed practical, the entire system would be implemented on an embedded platform.

1.4 General Approach to the Problem

In order to design a suitable system for the classification of EMG signals, one must realize that EMG signals are biological signals and have a lot of noise and variability associated with them. These characteristics make classification difficult as the EMG signal for the same gesture will never be identical. Many of the current approaches employ artificial neural networks to help resolve this issue. Neural networks have a distinct ability to model very complex relationships between the inputs and outputs of a system. This project makes use of the support vector machine (SVM) as the main classification algorithm, which has become a very popular tool for tasks involving classification.

The classification system will be composed of two modes: training and classification. To configure the system properly, it has to be trained using a data set gathered from the user. This is then analyzed to set the parameters for the SVM to be able to classify the myoelectric sigmals.

According to recent literature, the two most common neural networks used for the classification of myoelectric signals are multilayer perceptron neural networks (MLP) and support vector machines (SVM) [3, 4, 5, 2, 6]. Both of these methods have demonstrated classification accuracies of over 90% [2, 6]. However, the research suggests that the SVM provides slightly better classification accuracy accompanied by shorted learning times [6]. Improvements to the typical SVM have been documented, however to keep the complexity of the algorithm mangeable in the prescribed timeline these improvements will be kept as extra features that may be implemented if time permits. Ideally, the project would make use of an SVM classification algorithm provided the algorithm is not too excessive for the embedded platform. Many of the previous implementations have made use of PC software to classify the myoelectric signals including MATLAB [5, 6] and LIBSVM [5, 2]. Since this software would be unavailable on an embedded platform, it would be impractical to use these tools to begin development. The implementation will have to be developed from scratch.

1.5 Scope of the Project

The scope of the project is mostly limited to the development and implementation of a system that is able to classify myoelectric signals. There is no attempt to improve on current theoretical approaches. The initial intention of the project was to design the EMG classification system entirely on an embedded platform, including both modes of operation training as well as classification. This would ensure a very practical solution where the operation of the device would require substantially less equipment and less procedures.

2

Literature Review

2.1 Electromyographic Control

The idea of being able to control artificial limbs with electromyographic signals goes back to the 1970s. At that point in time, the typical method of controlling an artifical limb was through an amplitude-level coding algorithm [7]. This type of controller would essentially require the patient to produce an almost exact amplitude within the EMG signal to achieve the desired movement of the prosthetic device. Needless to say, this was a very simple approach to a very complicated problem. Using this method of control, it would require a great deal of time to train the patient to produce an EMG signal with a specific amplitude. Even if training is more or less successful, the results may not always be reproducible. EMG signals are biological signals and as a result they are modeled as random processes. This implies that an EMG signal that is generated when a patient attempts to open their wrist is unique.

Because of the obvious need for less improvised methods, researchers began to focus on more rigorous statistical analysis to be able to classify EMG signals, which included autoregression models as well as autoregressive-moving-average (ARMA) models [7]. In an ideal setting pure statistical analysis could yield good results being able to extract features that are common to all signals of a particular set. However, in a practical everyday situation the performance would decline, which could be easily attributed to increased noise in the acquisition portion of the system. This is one major disadvantage of using EMG signals for control purposes. The fact that surface electrodes are required to acquire the signal may make the signal more sensitive to the movements of the patient. Muscle fatigue is another reason the performance would decline. It is common knowledge that if one uses a particular group of muscles for an extended amount of time, they will fatigue. Once fatigue sets in, the same group of muscles will not be able to produce the same level of force and in turn the

EMG signal will diminish.

Ideally, in order to minimize the effect of all these fluctuations a system should incorporate the concept of pattern recognition that would also be able to recognize slight generalizations of the pattern. Pattern recognition was not suggested until 1972 [7]. However, due to the limited processing power and the complexity of the algorithms, these approaches were not typically used. It was not until 1993 that pattern recognition was successfully used to classify EMG signals [8]. *Hudgins et al.* [8] were able to use an artifical neural network to classify EMG signals.

2.2 Neural Networks

A neural network is a computational algorithm or machine that attempts to model the human brain. Its structure consists of an extremely parallelized network of "neurons". Each of the neurons, or computational units, receives a multiple number of input signals and produces a single output by performing a simple computation. In order to achieve the phenomenon of learning as the human brain does, the weight of the connection, or synapse, between neurons is altered through a learning algorithm. Figure 2.1 shows an example of a two layer neural network composed of a hidden layer and an output layer where each of the arrows in the figure represents a synapse with its own weighting. The input of the neuron j can be modeled as

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

where $y_i(n)$ represents an input to neuron j and $w_{ji}(n)$ represents the weighting between the input and neuron j.

Neural networks are remarkable as they are theoretically able to learn any task. Its main benefits include its parallel distributed structure and its ability to generalize [1]. This generalization can be demonstrated by the fact that a neural network can produce a reasonable result based on an input that it has never come across during its training.

The neural network structure used in [8] was a simple two-layer network that was trained using the back propagation algorithm. The back propagation algorithm is a type of supervised learning, in that it requires a "teacher" to decide whether the response of a neural network to a certain input is correct [1]. The idea behind training this type of neural network is to reduce the total error of the system. Once the neural network produces an output, the error between the output and the desired output is calculated. This error is then transferred to the preceding layer of neurons in order to recalculate the weightings of those synapses.

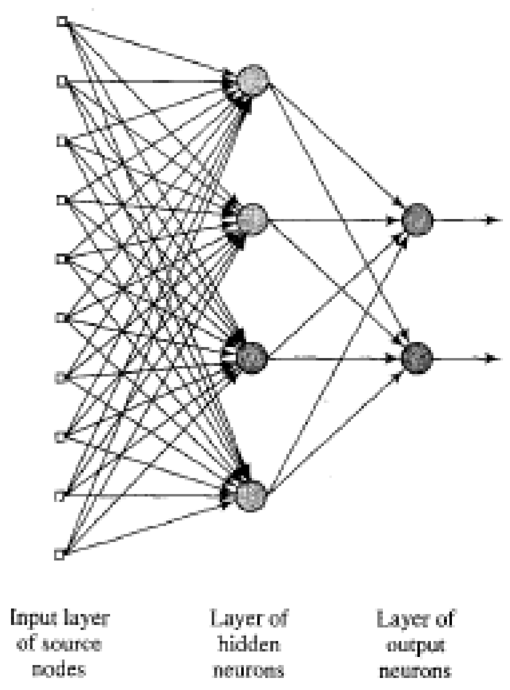


Figure 2.1: Neural network with one hidden layer and one output layer [1]

The process repeats until the error minimized enough to yield acceptable results.

Although the result of the training may produce acceptable results, the process may take a very long time because of the stochastic nature of the learning algorithm. Since the back propagation algorithm uses an instantaneous estimate of the error, it follows a random path and eventually converges to a minimum error [1]. However, this minimum error is typically a **local** minimum but there is no guarantee that this error is a global minimum. Furthermore, if the neural network is trained on a large set of training data, it may simply "memorize" the training data. The neural network would be able to recognize very specific patterns in the training data and may lose its ability to recognize slight variations from the norm. Essentially, this would decrease its ability to generalize and would lead to overfitting [1].

With this type of neural network the number of input, output and hidden nodes is decided by the problem at hand. There is no exact method of determining the optimal number. A general guideline states that the hidden layer should be as small as possible in order to minimize the complexity of the network, but it must be large enough to be able to adequately map the inputs to the outputs. This is an important issue as the neural network is not theoretically proven to be the optimal solution to the problem.

2.3 Feature Extraction

At first glance, it would seem reasonable to use the entire raw EMG signal as an input to the neural network. This would require one input node for every data point in the EMG signal. The obvious disadvantage would be the decrease in performance because of the large complexity of the input stage of the neural network. In addition, because of the variability in the EMG signals that are produced, using the raw EMG signal could lead to poor classification performance since each of the data points would have a large variance from signal to signal.

The inputs to the neural network in [8] consisted of a certain set of features that characterized the overall EMG signal. These features included: mean absolute value, mean absolute value slope, zero crossings, slope sign changes and waveform length. Although the EMG signal has a very large variance, these statistics attempt to offer enough stability so that they can adequately characterize a particular type of contraction over a variety of EMG signals. Using this approach *Hudgins et al.* [8] were able to achieve a 90% classification rate.

Feature extraction is a very interesting area of research. Many mathematical tools and models have been developed to characterize waveforms, however choosing the appropriate features is typically only based on empirical results. In comparison with [8], better results have been achieved using features based on wavelet transforms and wavelet packet transforms [9]. Using a wavelet packet transform feature set, the four channel continuous classification system described in [9] only yielded a 3.5% error rate when discriminating between six classes.

2.4 Support Vector Machines

Ever since *Hudgins et al.* [8] developed their pattern recognition based multifunctional myoelectric control system, many other types of algorithms have been employed to classify EMG signals, such as, linear discriminant analysis, radial basis function neural networks, fuzzy networks, Gaussian mixture models and hidden Markov models [2]. Recently, some researchers have begun to employ support vector machines to EMG analysis [5, 6, 2, 10].

Support vector machines (SVM) have only recently begun to receive increasing attention even though the theory was originally proposed by Vapnik in 1992 [11]. This attention is due the fact that SVMs have some interesting properties that will be discussed after a brief introduction to SVMs. SVMs can be described as binary classifiers, which determine whether a sample belongs to one class or another. They make use of the concept of structural risk minimization which provides a trade off between the empirical error and the model's complexity [1].

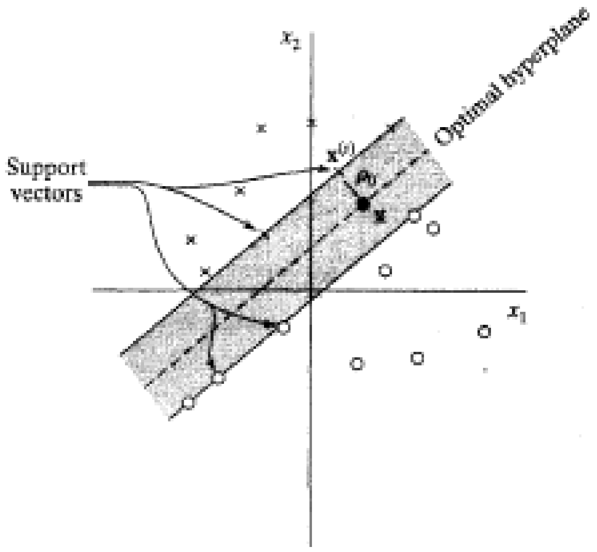


Figure 2.2: Optimal hyperplane with linearly separable data [1]

An integral part of an SVM is the construction of a decision surface such that the distance between positive and negative examples is maximized. Since the input vectors are elements of R^n, the decision surface is a hyperplane in a higher dimension. For simplicity we will first discuss linearly separable data, i.e. data that can be separated with a simple line [1]. In this case, the optimal hyperplane can be described by

$$\mathbf{w}^T\mathbf{x} + b = 0$$

where $\mathbf{w}$ represents a weight vector and $\mathbf{x}$ represents an input vector. Figure 2.2 shows an example of an optimal hyperplane with linearly separable data in two dimensions. Once the optimal hyperplane is determined a particular set of input vectors are used to define this hyperplane, which are called support vectors. As seen in Figure 2.2 these vectors are typically those input vectors that lie closest to the optimal hyperplane. For any new samples $\mathbf{x}_i$, classification is then performed based on the following conditions

$$\begin{array}{lll} \text{if} & \mathbf{w}^T\mathbf{x}_i + b \geq 0 & y_i = +1 \\ \text{if} & \mathbf{w}^T\mathbf{x}_i + b < 0 & y_i = -1 \end{array}$$

where $\mathbf{x}_i$ is the i^{th} sample and y_i is the i^{th} output.

The concept of an SVM is simple and very intuitive. However, difficulties arise when

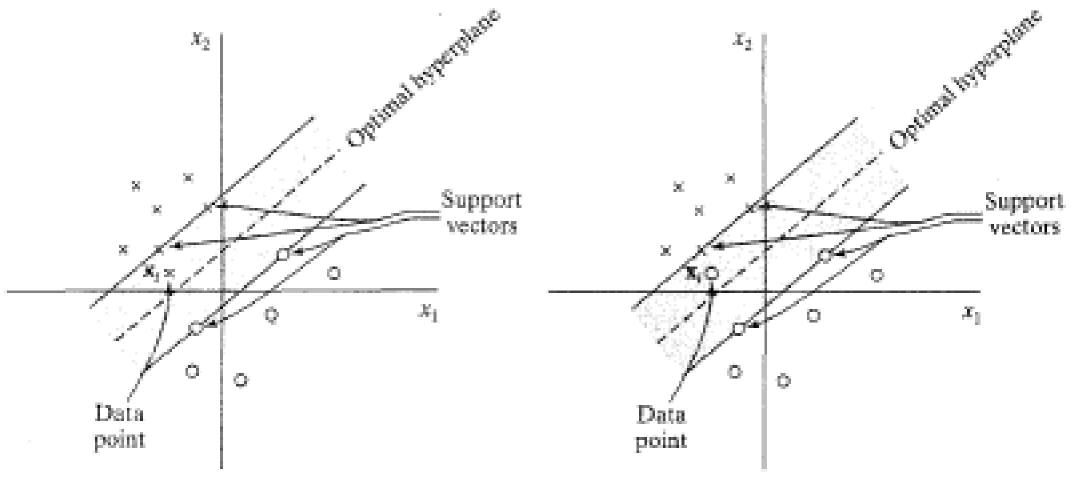


Figure 2.3: Optimal hyperplane with nonseparable data [1]

the data is nonseparable. Figure 2.3 illustrates the issue where a data point falls within the margin of separation, which is the distance between the optimal hyperplane and the support vectors, or the data point falls on the other side of the hyperplane. SVMs take this into account by means of a slack variable ξ_i which accounts for the deviation of a data point from its ideal position. Thus, the general form of an SVM can be succintly summarized by the following optimization problem [12]:

For a set of labeled pairs $(\mathbf{x}_i, y_i), i = 1, ..., l$ *where* $\mathbf{x}_i \in R^n$ *and* $y \in \{1, -1\}$ *an SVM is defined by the solution to:*

$$\begin{aligned} min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \textit{such that} \quad & y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\, \xi_i \geq 0 \end{aligned}$$

The optimization problem includes a C parameter, which is a user-defined parameter. This parameter is a penalty parameter of the error which controls the balance between the complexity of the SVM and the number of nonseparable points. With this parameter, the user can control the SVM's ability to generalize. This problem is then solved using quadratic optimization.

As there is an issue with the separability of data, there is another element in the structure of an SVM. SVMs employ a nonlinear kernel function that tranforms the input vector defined in the input space to a vector in a higher dimensional feature space. In doing so there is a stronger probability that the input vectors are more separable.

After the optimal hyperplane is constructed, the operation of an SVM is rather simple. The input vector is mapped into the feature space. The SVM then determines where the vector is located with respect to the hyperplane and that consists of its decision. The unique advantage of SVMs is their ability to generalize. By incorporating the deviation of input vectors as a variable in its optimization algorithm, SVMs can account for some variability of the input vector which makes it a good candidate for classifying EMG signals.

2.5 Support Vector Machines in EMG Classification

A number of researchers have made use of support vector machines with a direct application to the classification of EMG signals [5, 6, 2, 10]. All of the implementations have reported accuracy rates above 90%, the highest being 96.76% [6].

Yoshikawa et al. [5] proposed a real time hand motion estimation method using EMG signals. To accomplish this, they used four surface electrodes placed on a patient's forearm to distinguish between seven hand motions: at rest, opening of the hand, closing of the hand, pronation, supination as well as wrist flexion and extension. The EMG signals were measured using a professional EMG measurement device (Personal-EMG, Oisaka Development Ltd.) and the rest of the signal processing and classification was performed on a PC. In order to perform the classification they used an SVM with input vectors composed of the integrated EMG signal, cepstrum coefficients which are used in speech analysis and regression coefficients. With this set of features, the classification rate ranged from approximately 87% to 92%.

Oskoei et al. [2] compared the classification ability of SVMs with linear discriminant analysis (LDA) and multilayer perceptron (MLP) neural networks. This article described many interesting observations that resulted from slightly modifying many of the parameters surrounding the calculation of the input vector. One of the experiments tested the accuracy of classifications that were performed on different lengths of EMG data ranging from 50 to 500 ms. The performance of various features or sets of features was evaluated. The results of the research indicated that longer segments of raw EMG data produced better classification accuracy. The SVM based classifiers achieved the highest classification accuracy at 95.5%, with the LDA classifier performing at 94.5%. A two layer MLP performed with similar accuracy to the SVM and LDA, whereas a one layer MLP lost 6% in accuracy.

Rekhi et al. [10] performed EMG classification using features determined through wavelet packet analysis and then singular value decomposition. The purpose of the singular value decomposition was to reduce the dimensionality of the coefficients produced by the wavelet packet analysis. This study attempts to classify six different hand motions similar to the previous articles and results in an accuracy rate of 96%.

Liu et al. [6] used a much more complex cascaded kernel learning machine which was composed of a generalized discriminant analysis algorithm as well as an SVM. It documents the superior performance of this classifier in comparison with other common neural networks including the k-nearest neighbor algorithm, MLP networks and SVMs. The cascaded kernel learning machine was able to achieve an accuracy of 93.5%. This research team moved further than the previous ones in that it implemented a digital signal processor based EMG classification system in order to demonstrate the practicality of the solution. The classification module was developed on the DSP TMS320C31 which performs a classification based on 1000 raw EMG data points sampled at 2.5kHz.

3

Statement of Problem

The goal of the group project is to develop a robotic prosthetic device that can be controlled using EMG signals. The main motivation for this device is the need for greater functionality in prosthetic devices as well as the need for better interfaces. The group project consists of three major systems: EMG data acquisition, EMG classification and the robotic prosthetic.

This project attempts to fill a void between man and machine, which addresses the seemingly simple issue of being able to autonomously determine the hand gesture a patient is performing from the corresponding EMG signal produced by the muscles of the patient. As has already been discussed, the EMG signals that are produced while performing a particular hand gesture are never identical which requires the use of a pattern recognition algorithm. The hand gestures that would be classified would consist of gross movements of the hand including: rest position, opening of the hand, closing of the hand, wrist flexion, wrist extension, wrist pronation and wrist supination.

For practical purposes, the goal for the control system of the prosthetic device is real time (or at least continuous) operation. This EMG classification system will need to be developed specifically to interface with an EMG data acquisition system as well as a robotic prosthetic hand. To meet these constraints and to be a practical solution, the system would inevitably have to be developed on an embedded platform. Other practical matters that should be taken under consideration are power consumption and the bulkiness of the physical equipment.

4

Methodology of Solution

4.1 Theoretical Development

4.1.1 Multiclass SVM

Based on the current literature, the most popular and the most successful algorithm used for EMG classification are support vector machines or derivations of them [5, 6, 2, 10]. As was described in the Literature Review, the accuracy rate of the EMG classification using an SVM is typically above 90%. The main issue that remains is the configuration of the SVM and the choice of the features that will be extracted from the EMG signal.

Since SVMs are binary classifiers, a single SVM would only be able to classify two classes of gestures. In order to classify six gestures, a multiclass SVM is required which can be accomplished in one of two ways. The first approach would involve solving the optimization problem involving data from all six classes. The second approach simply builds the multiclass SVM out of a combination of binary SVMs. The combination of binary SVMs is much simpler and does not hinder the performance of the classification [2].

In order to implement the multiclass SVM using binary SVMs, there are two possible schemes that can be utilized: "one against all" (OAA) or "one against one" (OAO) [2]. For n classes, OAA uses n binary classifiers where each classifier is trained to distinguish between one class and the remainder of the classes. OAO requires $\frac{n(n-1)}{2}$ binary classifiers, where each classifier distinguishes between a pair of classes. In OAO, the final classification is based on a voting mechanism, where the outputs of all the classifiers are added together. The class with the most votes is the final output. According to [2], both schemes show more or less the same performance. However by comparing all pairs of classes, the OAO scheme calculates a better measure of the probability of each class. The EMG classification system implemented in this project using the OAO scheme.

4.1.2 SVM Kernel

An SVM uses a kernel in order to map the input data to a feature space. There are three commonly used kernels [12]:

$$\begin{aligned}
\text{linear:}\ K\left(\mathbf{x}_i, \mathbf{x}_j\right) &= \mathbf{x}_i^T \mathbf{x}_j \\
\text{polynomial:}\ K\left(\mathbf{x}_i, \mathbf{x}_j\right) &= \left(\gamma \mathbf{x}_i^T \mathbf{x}_j + r\right)^d, \gamma > 0 \\
\text{radial basis function:}\ K\left(\mathbf{x}_i, \mathbf{x}_j\right) &= e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \gamma > 0 \\
\text{sigmoid:}\ K\left(\mathbf{x}_i, \mathbf{x}_j\right) &= \tanh\left(\gamma \mathbf{x}_i^T \mathbf{x}_j + r\right)
\end{aligned}$$

In terms of selecting a kernel function to use with the SVM, there are no methods that can determine what kernel function should be used for a particular application. According to [12], the radial basis function (RBF) kernel should be a first choice. One reason being that for certain parameters C and γ the RBF kernel can behave like sigmoid and linear kernels. Another reason to use the RBF kernel is that there are less difficulties with mathematical computations. The output of the RBF kernel ranges between 0 and 1, whereas a polynomial kernel may approach infinity or 0.

4.1.3 Feature Extraction

As was mentioned in the Literature Review, feature extraction is not an exact science. There is still much work to be done in this area to determine the features that best represent EMG signals. Luckily, the literature has revealed a few studies that attempt to determine the features that produce the best results for EMG classification. *Zardoshti-Kermani et al.* [13] present an investigation into the class discrimination, robustness as well as computational complexity of a number of features. Using the Davies-Boulin index whihc is a measure of cluster separability as well as the k-nearest neighbour algorithm, they were able to compare: the integral of the absolulte value, zero crossings, variance, Willison amplitude, v-order detector, log detector, an autoregressive model as well as the histogram of the EMG signal. The analysis indicated that the histogram of the EMG signal had the best performance. *Huang et al.* [14] compare a similar list of features: the integral of the EMG signal, waveform length, variance, zero crossing, slope sign changes, Willison amplitude, a 4th order autoregressive model and a histogram of the EMG signal. The best results were obtained using the autoregressive model and the histogram of the EMG signal, however the analysis was done using a k-nearest neighbour algorithm. Using the results from [14], *Liu et al.* [6] achieve very good results with features based on the 4th order autoregressive model and a histogram of the EMG signal using the cascaded kernel learning machine.

It should be noted that when determining the types of features to extract from an EMG signal, one must seriously account for computational complexity in addition to its ability to help separate the data. Furthermore, as the project's focus is on the design and implementation of an EMG classification system and not the development of new feature extraction methods, the chosen feature extraction methods will be based on those that have been proven to be effective. From the literature, the two best candidates are the autoregressive model as well as the histogram of the EMG signal.

Autoregressive Moving Average Model

The analysis of EMG signals using an autoregressive moving average model (ARMA) dates back to 1975 [7]. In order to model an EMG signal, a fourth order model is adequate [13]. Using a least squares algorithm over a certain time window, the calculation of the model of order p can be described by the following equations

$$\mathbf{a}_i = \mathbf{a}_{i-1} + \mathbf{P}_i \mathbf{X}_i \left(x_i - \mathbf{X}_i^T \mathbf{a}_{i-1} \right)$$
$$\mathbf{P}_i = \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{X}_i\mathbf{X}_i^T\mathbf{P}_{i-1}}{1 + \mathbf{X}_i^T\mathbf{P}_{i-1}\mathbf{X}_i}$$

where $\mathbf{X}_i = \left[x_{i-1}\, x_{i-2} \ldots x_{i-p}\right]^T$, $\mathbf{a}_i$ is the vector of the ARMA coefficients, $\mathbf{a}_{i-1}$ contains the previous ARMA coefficients, $\mathbf{P}_i$ is a p by p matrix. Before any calculations are begun, $\mathbf{P}_0 = \mathbf{I}$ and $\mathbf{a}_0 = 0$. These equations are used to perform a recursive calculation over all the data points within a certain window, updating $\mathbf{a}_i$ and $\mathbf{P}_i$ on each pass.

EMG Histogram

The EMG histogram draws on a simple observation. As a muscle contracts, the signal begins to deviate from its resting position. Measuring the frequency with which the EMG signal reaches multiple amplitude levels proves to be an effective tool to characterize an EMG signal. The histogram is implemented in a similar fashion as has been documented in the literature [13]. It separates a voltage range that is symmetric around 0 V into 10 bins.

4.2 EMG Classification System

The overall design of the EMG classification system can be seen in Figure 4.1. There will be three channels to record EMG signals at 3 postions on the upper extremity. Feature extraction will be performed on each of the 3 channels to produce an input vector. The system will consist of two modes of operation: a training phase and a classification phase.

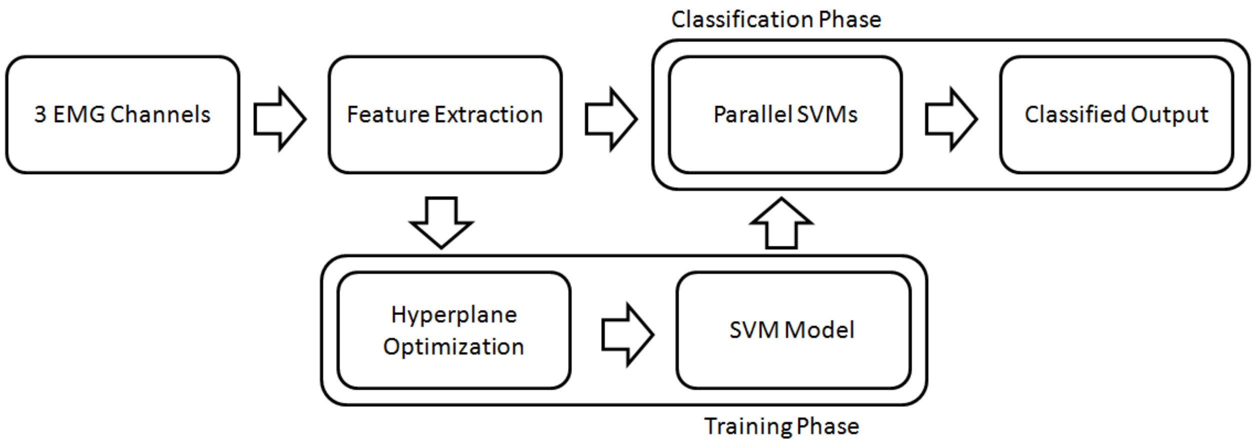


Figure 4.1: Generalized block diagram of the EMG Classification System

During the training phase, each binary SVM will be trained using control data provided by the patient which will result in a model. The model will then be used during the classification phase to continuously classify new EMG signals provided by the patient.

5

Design Procedures and Experimental Procedures

5.1 Hardware Design

During the initial stages of the design of the EMG classification system, the hardware requirement seemed to be any ordinary processor that could be interfaced with an EMG data acquisition system as well as a robotic prosthetic. In the most general sense, the system would be required to perform a large amount of data manipulation, which would be implemented entirely in software. Before looking into the various types of microcontrollers and microprocessors that would be suitable for the task, it was suggested to me by Phillip Kinsman to develop an FPGA-based implementation. The main reason for the suggestion stems from the fact that the EMG acquisition system would be implemented on this FPGA. The interface between these two parts of the overall project would be much smoother and it would reduce the need for another clunky piece of equipment. The FPGA used to develop the EMG classification system was the Altera DE2-70 which can be seen in Figure 5.1.

FPGA stands for field-programmable gate array which is essentially an integrated circuit that is designed to be reprogrammable. FPGAs contain a large amount of logic blocks and memory elements that can be configured together to produce useful logic operations. In order to program an FPGA, a hardware description language (HDL), like Verilog, is used which helps with reducing the complexity when designing a digital system. There are also many advantages to using an FPGA. Depending on the type of application, very fast solutions can be developed using custom hardware. In implementing custom hardware, the solution is tailored specifically to the needs at hand and eventually result in less power consumption when compared with a general purpose digital signal processor. In comparison

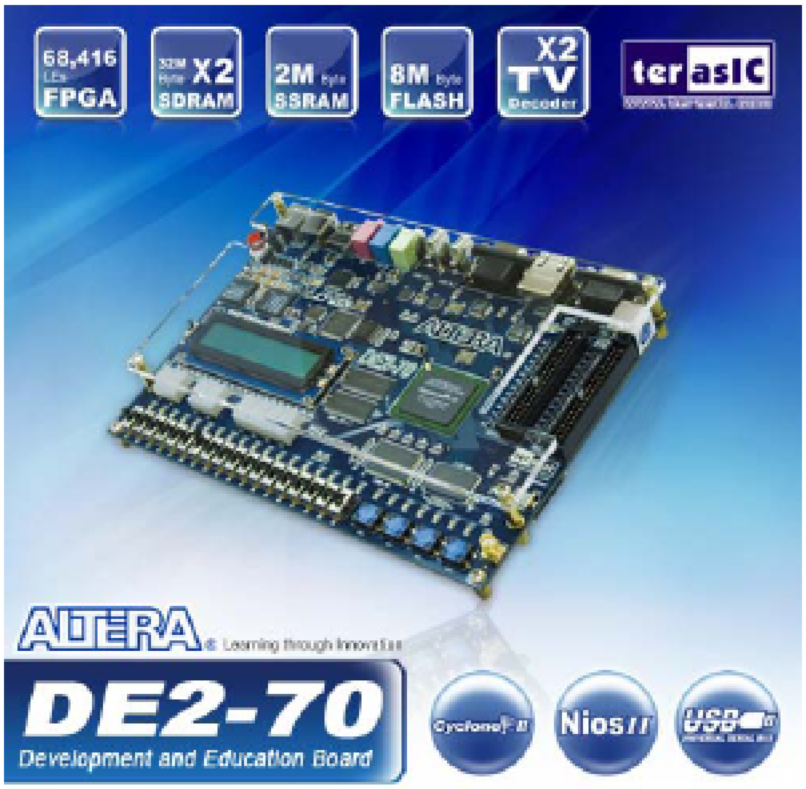


Figure 5.1: Altera DE2-70 FPGA Development Board

with a microprocessor, an FPGA runs at a much slower clock rate however operations can be performed in parallel whereas a microprocessor can only execute instructions serially. FPGAs can be reprogrammed. In a manufacturing process, this would be very desirable as any patches could still be applied to products in the field.

To simplify development it was initially decided to use a soft core processor on the FPGA to run the classification algorithm. The soft core processor is called Nios II which is a 32-bit embedded processor architecture that can be placed on an FPGA. This would allow development in C and avoid the need to develop the complex algorithm in Verilog.

5.2 Software Design

The bulk of the software implmentation of the EMG classification system lies within the SVM. As the project's scope does not include the development of a more efficient implementation of an SVM, an open source implementation was chosen as a starting point. Two implementations were found that were most commonly used in the literature: LIBSVM [15] and SVMLight [16]. SVMLight was chosen simply on the fact that it was written in C which would correspond directly with the development enviroment of the Nios II processor.

The SVMLight implementation produced two exectuables: svm_learn and svm_classify. svm_learn was used to train the SVM based on samples in a specified file and it output a model file that could be used in the classification process. svm_classify performed classifications based on samples in a file and the model file produced by svm_learn. In order to make the system more automatic these executables were incorporated into a larger multiclass SVM.

The initial protoyping was done on a PC using Microsoft's Visual Studio. In order to classify 6 gestures, using the OAO method this would require 15 binary SVMs, each of which needs to be trained on a particular data set. The training algorithm was set up in such a way that it asked the user for an input file, containing control EMG data from the patient, for each gesture. This control data would have been captured using an EMG data acquisition system while the patient was performing a certain gesture. Since these files would contain the raw EMG data, features would have to be extracted for each EMG signal. The input vector from each channel would be composed of 14 components, 10 from the EMG histogram (HEMG) and 4 from the ARMA model. Therefore, since the initial design was to accomodate 3 EMG channels, the entire input vector would be composed of 42 components. Once the raw EMG data is processed for all the samples describing one gesture it is stored in memory until all the gestures have been processed. Then a training data file is created for each SVM containing the specific input vectors of the two gestures that it classifies. For example, the third SVM may be charged with the responsibility of classifying between wrist flexion and wrist extension. The training data file would contain all the input vectors describing wrist flexion and wrist extension that were calculated in the previous stage. All that remains is the actual training of the SVMs.

In order to get better classification performance, the training algorithm requires the user to specify two parameters for each SVM: γ and C. C was discussed in the section describing SVMs and γ is a parameter used in the kernel function. Since the data set for each SVM is different, each of these parameters can be adjusted in such a way to help minimize the classification error. This is automated by using a grid search algorithm outlined in [12]. A 4x4 grid is created where each node has a corresponding value of γ and C. Each of parameters are initialized to values between 2^{-8} and 2^{8}. Once the grid has been created, a particular SVM is trained sequentially using the parameters specified by each node in the grid. The SVMLight implementation allows the calculation of an error estimate, which is based on a leave one out approach. This approach can be better illustrated with a thorough example. Supposing the training data consists of 5 input vectors, to determine the accuracy of the model, one of these input vectors is left out and the model is retrained using the remaining 4 input vectors. Once the model is retrained, it is used to classify the input vector that was left out. This process is repeated for each of the input vector. Since the desired outcome is known for each classification a corresponding total error can be calculated. Of course this is not an ideal measure of the error, but it does provide a better understanding of classification ability of the model. After all the error estimates are calculated, the grid search is refined within the neighbourhood of the node with the highest accuracy. The parameters, showing the most accurate results after this refinement, are chosen as the final parameters and the SVM is trained one last time. As soon as training is completed, a model file is created which is used in the classification phase.

A note should be made regarding the run time of this algorithm. The leave one out method was chosen to estimate the error as it was provided in the implementation. [12] provides an alternative method called cross validation. It is similar to the leave one out method, however it divides the training data set into large subsets and computes the error while leaving out those subsets instead of individual input vectors. As this development was only concerned with prototyping, it was not an issue at this point in time.

The classification phase is more straightforward. The user is prompted for an input file containing the raw EMG signal to be classified. The features are extracted from the signal and the input vector is processed by the SVM model that was created during the training phase. The result is a number between -1 and 1 which indicates where the new sample lies in comparison to the decision surface. This process would be repeated for all of the SVMs.

All of the outputs are then tabulated to vote on a final output. This output would then be used to drive the robotic prosthetic device.

5.3 Runtime Measurements

Since the system was being developed for an embedded platform, a lot of time was spent measuring the various system requirements during the development process. The main runtime measurements were made with respect to memory allocations as well as processing time.

The results of these measurements indicated that the training phase of the SVM required a large amount of memory. This is due to the fact that SVMLight uses a fast kernel caching algorithm, which is meant to speed up training times. The default setting for this kernel cache is 40MB whereas the absolute maximum amount of fast memory (SSRAM) available on the FPGA was 2MB. It should be noted that although there were 2 blocks of 32MB of SDRAM on the development board, we did not have the memory controllers to make use of these blocks at this point in time. This memory requirement could be scaled by simple decreasing the kernel cache to less than 2 MB at the price of a decrease in training time. When it comes to processing time, it is a rather lengthy process. As the training data can vary from training run to training run, it is never the same length of time. The training time can range anywhere from a few minutes to over half an hour, depending on the data set and the choice of C and γ parameters.

The classification phase, on the other hand, performed much better. The memory usage depended greatly on how many support vectors were included in the model file, since one of these files was stored in memory at all times during classification. The processing time took slightly longer than 1s. Although this is an acceptable result for offline classification, it presents a slight problem when it is meant to perform classifications continuously. Ideally, a maximum delay of a few hundred milliseconds would be preferred to 1s.

An important issue that arose at this point was the almost overwhelming complexity of the algorithm. The classification system would have to be ported from a PC running a Core 2 Duo CPU at 2.1GHz to an FPGA development board where the clock runs at 50MHz. In using the FPGA to implement the classification algorithm, the main objective was to reduce the number of components that would make up the entire system and to streamline the interface between all three portions of the project. By planning to use the Nios II processor, all the instructions would essentially be processed serially leading to a 42x increase in processing time. Since this was unacceptable for an embedded system, design alterations had to be made.

5.4 Design Alterations

The results from the runtime measurements were not too encouraging. A large portion of the algorithm had to be redesigned in terms of implementation. This was done by considering each of the two modes of operation separately.

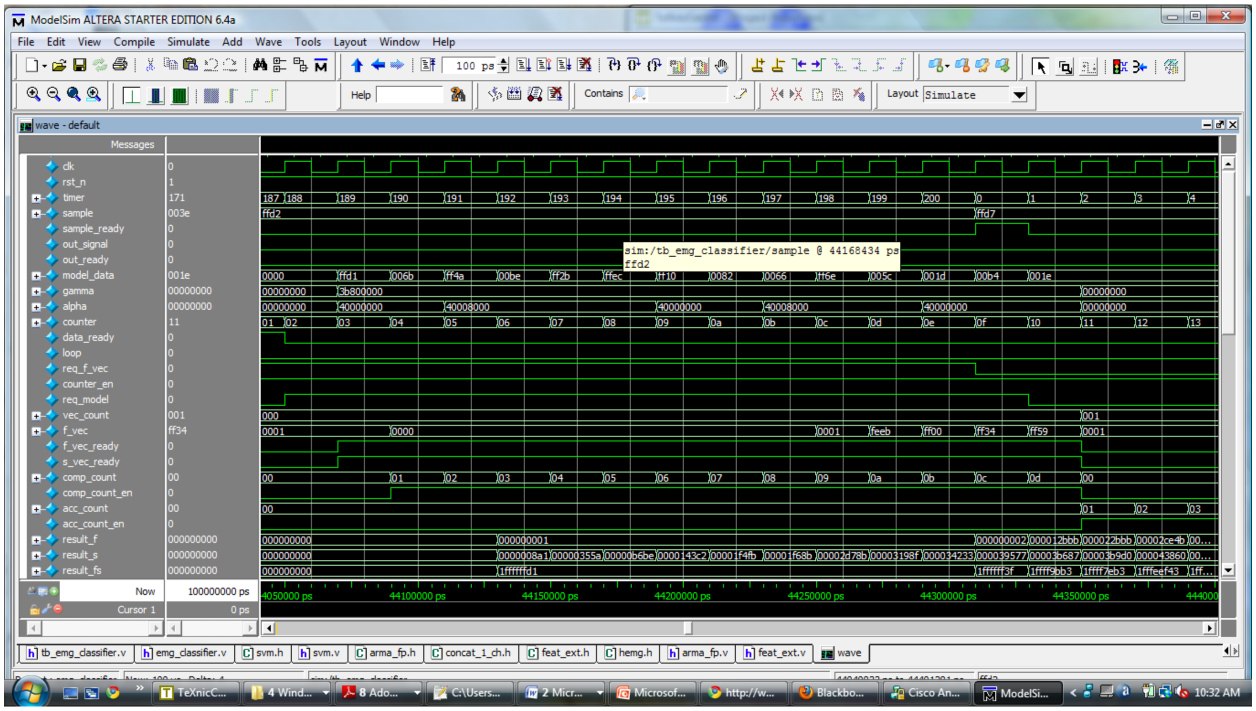
It was inevitable that the training phase would require a longer amount of time no matter where or how it was implemented. Since the training algorithm is very involved and very complex it would require a considerable amount of time and skill to develop a more efficient solution on the embedded platform. From the point of view of the patient in a practical situation, they would probably not sit around and wait while the device was training. This phase could be completed over a longer period of time. The current implementation, although not on an embedded platform, satisfied this point. In order to perform the training on a PC, we required a method to transfer the data from the development board. The board has almost all of the standard input and output interfaces, including an SD Card as well as USB. Since an SD Card controller was available for the DE2-70, it was decided that the data could be transferred using an SD Card.

The classification phase posed a larger issue as processing time was a crucial component to the successful functioning of the system. A closer look at the classification algorithm revealed a somewhat straightforward computation, which could possibly be implemented using custom hardware. Taking advantage of the strengths of the FPGA, the algorithm could be implemented much more efficiently. Unfortunately, this would require the complete redevelopment of the entire classification phase.

Other issues outside of the project also required the scaling back of certain objectives. Being able to mimic six different gestures in a robotic prosthetic proved to be a difficult task. Furthermore, developing a prototype for the EMG data acquisition front end was also rather challenging. There was only enough time to build one data acquisition board. As a result, one EMG channel would not provide enough information to classify six different hand gestures. Therefore, to satisfy a proof of concept the classification was limited to 2 gestures: opening of the hand and closing of the hand. This would only require one SVM

5.5 Custom Hardware Design

The solution to the runtime problems was to design custom hardware for the classification phase using Verilog. A major obstacle was my complete unfamiliarity with the language as well as unfamiliarity with custom hardware design. The general outline of the design did not change. Feature extraction would be performed on the raw EMG signal producing an input



Messages	
clk	0
rst_n	1
timer	171
sample	003e
sample_ready	0
out_signal	0
out_ready	0
model_data	001e
gamma	00000000
alpha	00000000
counter	11
data_ready	0
loop	0
req_f_vec	0
counter_en	0
req_model	0
vec_count	001
f_vec	ff34
f_vec_ready	0
s_vec_ready	0
comp_count	00
comp_count_en	0
acc_count	00
acc_count_en	0
result_f	000000000
result_s	000000000
result_fs	000000000
Now	100000000 ps
Cursor 1	0 ps

```
sim:/tb_emg_classifier/sample @ 44168434 ps
ffd2
```

Figure 5.2: Simulation of the EMG classifier custom hardware block

vector. The input vector would be classified by the SVM based on the model generated in the training phase and the SVM would produce a single output.

In designing the custom hardware, a lot of simulations were performed using ModelSim. A simulation of the EMG classifier can be seen in Figure 5.2. This proved to be a very useful tool in designing and debugging the various blocks.

Once the implementation of the hardware modules was complete, the compilation showed that a surprising amount of resources were required for the EMG classifier. The entire classifier, including the feature extraction modules, consumed about 15% of the FPGA, with the ARMA module consuming about 10% on its own.

The processing time of the custom hardware implementation was very fast. The raw EMG data samples were supplied at 10kHz, meaning one sample arrived every 5000 clock cycles. The custom hardware implementation allowed the feature extraction to occur as soon as the samples arrived. The HEMG module took approximately 3 clock cycles to update its internal registers whereas the ARMA module took over 200 clock cycles for a single iteration of the algorithm outlined earlier. One classification was based on 1024 raw EMG samples which turns out to be approximately 100ms, meaning that one classification would occur every 5 million clock cycles. In the digital world, this is plenty of time to perform any sort of calculation. One single pass through the SVM module took approximately 83 clock cycles,

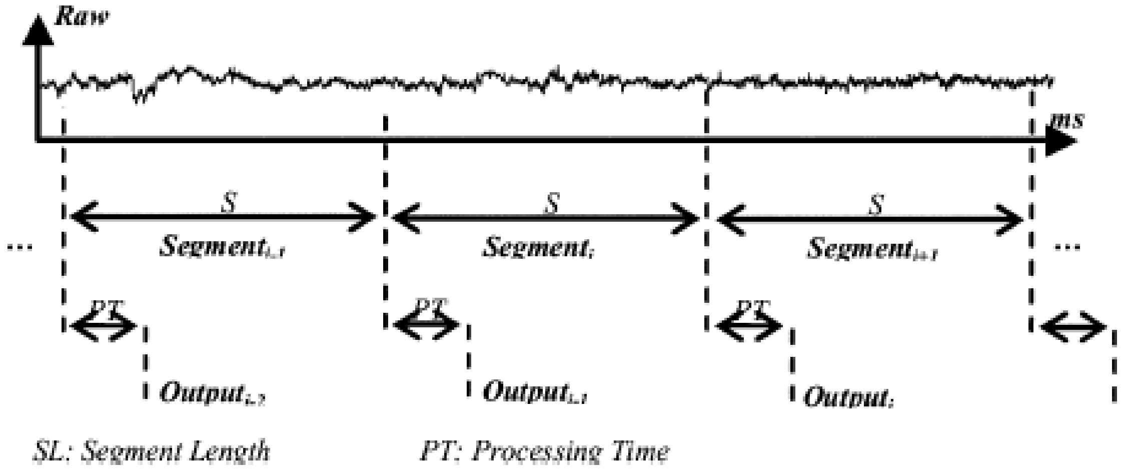


Figure 5.3: Diagram of the segmentation of the EMG signal for classification [2]

with an overlap of about 30 clock cycles between subsequent inputs. The processing time of this module depends heavily on the amount of support vectors that the model contains. For 400 support vectors this process takes approximately 17000 clock cycles. Figure 5.3 shows the operation of the EMG classifier in a continuous setting.

Thus with these processing times the classification is very fast, with the limiting factor being the length of the EMG signal window. A 100ms window allows a maximum of 10 classifications per second.

With this approach it would be very simple to extend this design to the 15 SVMs required to classify 6 gestures. All of the processing could be done in parallel, without any additional time added. The only problems would arise with the amount of resources.

6

Results and Discussion

Overall the results of the implementation of the EMG classification system were rather satisfactory. A functional demonstration of the prototype system demonstrated the ability to continuously classify the EMG signals recorded from the forearm of a patient. Based on qualitative observations, the response of the classification seemed a little delayed and not all classifications were made correctly. Since much of the overall system has been developed from scratch, it would be difficult to judge where the issues would reside.

The actual accuracy rate during online operation is unknown. Since the classification occurs in real time, it would be very difficult to implement something that keeps track of the classification error. It would at least require the answer to the question “Was that classification correct?”, which could only be provided by a human being. However, using the original classification algorithm developed on the PC, I was able to simulate the classification and determine an accuracy rate of approximately 82%. For a first attempt at EMG classification, I believe this is an acceptable result. Figure 6.1 shows a sample EMG signal recorded during the closing of the hand using the data acquisition system. It can be seen in the figure that there is some drift in the signal and still contains a lot of noise. The quality of the signal may be a factor in the classification accuracy of the SVM, since the SVM may be noticing the similarities between the noise patterns as opposed to the differences in the EMG signals.

During the operation of the system, I was able to notice that the electrodes that were being used, seemed to stop working after a short period of time. This obviously affected the signal as time passed by, since the signal was much more degraded.

Not all of the initial objectives were met, however much was learned in the process. The initial intention to use 3 EMG data channels as well as the ability to classify 6 different hand gestures proved to be far too difficult to implement within the time frame that was available. The main objective of the project was to develop an EMG classification system for an embedded platform and this goal was achieved.

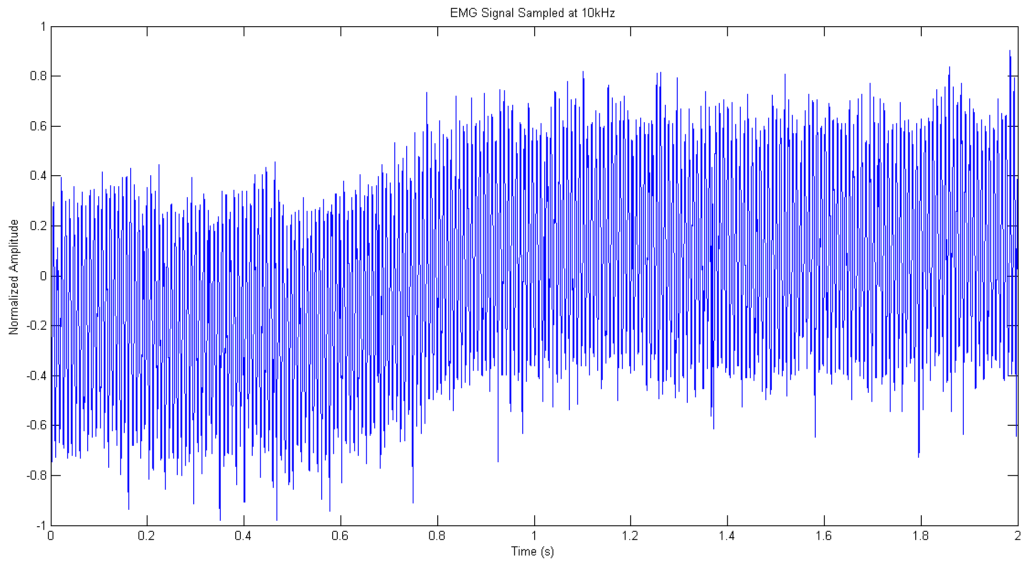


Figure 6.1: A sample EMG signal during closing of the hand

7

Conclusions and Recommendations

The project has succesfully demonstrated that electromyographic control on an embedded platform is achievable. With more time and experience, I am confident that this system would be able to achieve the original goal of the satisfactory classification of 6 differents hand gestures.

There are always many improvements that could be made. It would be interesting to see if the implementation of the training algorithm on the FPGA would be feasible. Furthermore, in order to be able to classify more gestures, more SVMs would be needed and as a result the modules would have to be much more efficient. Since there are so many clock cycles where the modules do nothing, it could be possible to extend the processing of these modules to just barely meet the time constraints. By doing this, each module could reuse its constituent multiplier and addition blocks, eliminating the waste of precious resources. It would also be interesting if this approach could be extended to the classification of the finer movements of the fingers.

8

Appendix

The source code used in this project can be found on the attached CD.

Bibliography

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2nd ed., 1999.

[2] M. A. Oskoei and H. Hu, “Support vector machine-based classification scheme for myoelectric control applied to upper limb,” *Biomedical Engineering, IEEE Transactions on*, vol. 55, no. 8, pp. 1956–1965, 2008.

[3] M. F. Kelly, P. A. Parker, and R. N. Scott, “Myoelectric signal analysis using neural networks,” *Engineering in Medicine and Biology Magazine, IEEE*, vol. 9, no. 1, pp. 61–64, 1990.

[4] K. Ito, T. Tsuji, A. Kato, and M. Ito, “Limb-function discrimination using emg signals by neural network and application to prosthetic forearm control,” *Neural Networks, IEEE International Joint Conference on*, vol. 2, pp. 1214–1219, 1991.

[5] M. Yoshikawa, M. Mikawa, and K. Tanaka, “Real-time hand motion estimation using emg signals with support vector machines,” *SICE-ICASE, 2006. International Joint Conference*, pp. 593–598, 2006.

[6] Y.-H. Liu, H.-P. Huang, and C.-H. Weng, “Recognition of electromyographic signals using cascaded kernel learning machine,” *Mechatronics, IEEE/ASME Transactions on*, vol. 12, no. 3, pp. 253–264, 2007.

[7] D. Graupe and W. K. Cline, “Functional separation of emg signals via arma identification methods for prosthesis control purposes,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 5, no. 2, pp. 252–259, 1975.

[8] B. Hudgins, P. Parker, and R. Scott, “A new strategy for multifunction myoelectric control,” *Biomedical Engineering, IEEE Transactions on*, vol. 40, no. 1, pp. 82–94, 1993.

[9] K. Englehart, B. Hudgin, and P. A. Parker, "A wavelet-based continuous classification scheme for multifunction myoelectric control," *Biomedical Engineering, IEEE Transactions on*, vol. 48, no. 3, pp. 302–311, 2001.

[10] N. S. Rekhi, A. S. Arora, S. Singh, and D. Singh, "Multi-class svm classification of surface emg signal for upper limb function," in *Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009. 3rd International Conference on*, pp. 1–4, 2009.

[11] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, (Pittsburgh), pp. 144–152, ACM Press, 1992.

[12] C.-C. C. Chih-Wei Hsu and C.-J. Lin, "A practical guide to support vector classification." Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2009.

[13] M. Zardoshti-Kermani, B. C. Wheeler, K. Badie, and R. M. Hashemi, "Emg feature evaluation for movement control of upper extremity prostheses," *Rehabilitation Engineering, IEEE Transactions on*, vol. 3, no. 4, pp. 324–333, 1995.

[14] H.-P. Huang, Y.-H. Liu, and C.-S. Wong, "Automatic emg feature evaluation for controlling a prosthetic hand using supervised feature mining method: an intelligent approach," *Robotics and Automation, Proceedings IEEE International Conference on*, vol. 1, pp. 220–225, 2003.

[15] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[16] T. Joachims, "Making large-scale svm learning practical," in *Advances in Kernel Methods - Support Vector Learning* (B. Schlkopf, C. Burges, and A. Smola, eds.), MIT-Press, 1999.

Vitae

Name: Philip Chrapka
Place of Birth: Hamilton, Ontario
Year of Birth: 1987
Secondary Education: Westdale Secondary School
Honours and Awards:
The W. Reymont Foundation Scholarship (2006, 2007, 2009)
The Stanislaw Smolinski Foundation Scholarship (2007,2008)
McMaster President's Award (2005)
Nortel Networks Entrance Scholarship (2005, 2005)
The PEO Foundation for Education in-Course Scholarship (2006)
Ontaio Scholar Award (2005), Carol Moule Memorial Award (2005)