2013

On Design and Implementation of an Embedded System for Neural-Machine Interface for Artificial Legs

Xiaorong Zhang
University of Rhode Island, dilys_84410@hotmail.com

ON DESIGN AND IMPLEMENTATION OF AN EMBEDDED SYSEM FOR NEURAL-MACHINE INTERFACE FOR ARTIFICIAL LEGS

BY

XIAORONG ZHANG

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER ENGINEERING

UNIVERSITY OF RHODE ISLAND

2013

DOCTOR OF PHILOSOPHY DISSERTATION

OF

XIAORONG ZHANG

APPROVED:

Dissertation Committee:

Major Professor Qing Yang

He Huang

Joan Peckham

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2013

ABSTRACT

According to limb loss statistics, there are over one million leg amputees in the US whose lives are severely impacted by their conditions. In order to improve the quality of life of patients with leg amputations, neural activities have been studied by many researchers for intuitive prosthesis control. The neural signals collected from muscles are electromyographic (EMG) signals, which represent neuromuscular activities and are effective bioelectrical signals for expressing movement intent. EMG pattern recognition (PR) is a widely used method for characterizing EMG signals and classifying movement intent. The key to the success of neural-controlled artificial limbs is the neural-machine interface (NMI) that collects neural signals, interprets the signals, and makes accurate decisions for prosthesis control.

This dissertation presents the design and implementation of a real-time NMI that recognizes user intent for control of artificial legs. To realize the NMI that can be carried by leg amputees in daily lives, a unique integration of the hardware and software of the NMI on an embedded system has been proposed, which is real-time, accurate, memory efficient, and reliable. The embedded NMI contains two major parts: a data collection module for sensing and buffering input signals and a computing engine for fast processing the user intent recognition (UIR) algorithm. The designed NMI has been completely built and tested as a working prototype. The system performance of the real-time experiments on both able-bodied and amputee subjects for recognizing multiple locomotion tasks has demonstrated the feasibility of a self-contained real-time NMI for artificial legs.

One of the challenges for applying the designed PR-based NMI to clinical practice is the lack of practical system training methods. The traditional training procedure for the locomotion mode recognition (LMR) system is time consuming and manually conducted by experts. To address this challenge, an automatic and user-driven training method for the LMR system has been presented in this dissertation. In this method, a wearable terrain detection interface based on a portable laser distance sensor and an inertial measurement unit is applied to detect the terrain change in front of the prosthesis user. The identification of terrain alterations together with the information of current gait phase can be used to automatically identify the transitions among various locomotion modes, and labels the training data with movement class in real-time. The pilot experimental results on an able-bodied subject have demonstrated that this new method can significantly simplify the LMR training system and the training procedure without sacrificing the system performance.

Environmental uncertainty is another challenge to the design of NMI for artificial limbs. EMG signals can be easily contaminated by noise and disturbances, which may degrade the classification performance. The last part of the dissertation presents a real-time implementation of a self-recovery EMG PR interface. A novel self-recovery module consisting of multiple sensor fault detectors and a fast linear discriminant analysis (LDA) based classifier retraining strategy has been developed to immediately recover the classification performance from signal disturbances. The self-recovery EMG PR system has been implemented on a real-time embedded system. The preliminary experimental evaluation on an able-bodied subject has shown that the system can maintain high accuracy in classifying multiple movement tasks while

motion artifacts have been manually introduced. The results may propel the clinical use of EMG PR for multifunctional prosthesis control.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Qing Yang, for his excellent guidance and continuous support during my Ph.D. study and for his motivation, enthusiasm, and immense knowledge that always inspire me. His mentorship has helped me all the time to grow both professionally and personally. I am truly fortunate to have him to be my advisor and mentor during the past six years.

Besides my advisor, I would like to thank the rest of my dissertation committee: Professors He Huang, Joan Peckham, Haibo He, and Philip Datseris for their time and help in participating in my comprehensive exam and dissertation defense, and reviewing the manuscript of this dissertation.

I want to thank Dr. He Huang for giving me the opportunity to work on this exciting multidisciplinary research project of neural-controlled artificial legs. Without her guidance and help, I would not be able to finish my dissertation. I would like to thank Dr. Haibo He, who has always encouraged me and given me a lot of insightful suggestions on my research and career. Much appreciation goes to Dr. G. Faye Boudreaux-Bartels. I have benefited a lot from her advice and help in my work and life.

I am also grateful to other faculty and staff members of the Department of Electrical, Computer, and Biomedical Engineering, in particular to Dr. Yan Sun, Dr. Resit Sendag, Dr. Steven Kay, Dr. Godi Fischer, Dr. William Ohley, Dr. Timothy Toolan, Mr. James Vincent, and Ms. Meredith Leach Sanders. I would also like to extend sincere thanks to my colleagues and friends for their support, help, and company during my Ph.D study.

I would like to thank my parents. They are always supporting me and encouraging me with their love and best wishes.

Finally, I would like to thank my husband, Quan Ding, for his love and support, always being the first subject in my research experiments, always being there cheering me up, and standing by me through good times and bad.

PREFACE

This dissertation is written in the manuscript format. It consists of five manuscripts organized as follows:

Manuscript 1:

Xiaorong Zhang, Yuhong Liu, Fan Zhang, Jin Ren, Yan Sun, Qing Yang, and He Huang, "On Design and Implementation of Neural-Machine Interface for Artificial Legs", published in *IEEE Transactions on Industrial Informatics*, 2012. 8(2): p. 418-429.

Manuscript 2:

Xiaorong Zhang, Qing Yang, and He Huang, "Design and Implementation of a Special Purpose Embedded System for Neural Machine Interface", published in *the proceeding of the 28th IEEE International Conference on Computer Design (ICCD'10)*, Amsterdam, Netherland, 2010. p. 166-172.

Manuscript 3:

Xiaorong Zhang, He Huang, and Qing Yang, "Implementing an FPGA System for Real-Time Intent Recognition for Prosthetic Legs", published in *the processing of the 49th Design Automation Conference (DAC'12)*, San Fransico, CA, 2012. p. 169-175.

Manuscript 4:

Xiaorong Zhang, Ding Wang, Qing Yang, and He Huang, "An Automatic and User-Driven Training Method for Locomotion Mode Recognition for Artificial Leg Control", published in *the proceeding of the 34th Annual International Conference of*

the IEEE Engineering in Medicine and Biology Society (EMBC'12), San Diego, CA, 2012. p. 6116-6120.

Manuscript 5:

Xiaorong Zhang, He Huang, and Qing Yang, "Real-Time Implementation of a Self-Recovery EMG Pattern Recognition Interface for Artificial Arms", submitted to *the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'13)*, Osaka, Japan, 2013.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

MANUSCRIPT 1

On Design and Implementation of Neural-Machine Interface for Artificial Legs

by

Xiaorong Zhang[1], Yuhong Liu[1], Fan Zhang[1], Jin Ren[2],

Yan Sun[1], Qing Yang[1], and He Huang[1]

is published in IEEE Transactions on Industrial Informatics, 2012. 8(2): p. 418-429

[1] Xiaorong Zhang, Yuhong Liu, Fan Zhang, Yan (Lindsay) Sun, Qing Yang, and He Huang are with Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, 02881, Email {zxiaorong, yuhong, fzhang, yansun, qyang, huang}@ele.uri.edu.

[2] Jin Ren is with VeloBit Inc, 80 Central St. Boxborough, MA 01719

Abstract

The quality of life of leg amputees can be improved dramatically by using a cyber physical system (CPS) that controls artificial legs based on neural signals representing amputees' intended movements. The key to the CPS is the neural-machine interface (NMI) that senses electromyographic (EMG) signals to make control decisions. This paper presents a design and implementation of a novel NMI using an embedded computer system to collect neural signals from a physical system - a leg amputee, provide adequate computational capability to interpret such signals, and make decisions to identify user's intent for prostheses control in real time. A new deciphering algorithm, composed of an EMG pattern classifier and a post-processing scheme, was developed to identify the user's intended lower limb movements. To deal with environmental uncertainty, a trust management mechanism was designed to handle unexpected sensor failures and signal disturbances. Integrating the neural deciphering algorithm with the trust management mechanism resulted in a highly accurate and reliable software system for neural control of artificial legs. The software was then embedded in a newly designed hardware platform based on an embedded microcontroller and a graphic processing unit (GPU) to form a complete NMI for real time testing. Real time experiments on a leg amputee subject and an able-bodied subject have been carried out to test the control accuracy of the new NMI. Our extensive experiments have shown promising results on both subjects, paving the way for clinical feasibility of neural controlled artificial legs.

1.1 Introduction

There are over 32 million amputees worldwide whose lives are severely impacted by their condition. This number is growing as the population ages and as the incidence of dysvascular disease increases. Over 75% of major amputations were lower-limb, with nearly 17% of lower-limb amputees suffering bilateral amputations [1]. Therefore, there is a continued need to provide this large and growing population of amputees with the best care and return of function possible.

With the rapid advances of cyber system technologies, it has in recent years become possible for high speed, low cost, and real time embedded computers to be widely applied in biomedical systems. The computerized prosthetic leg is one prominent example, in which motion and force sensors and a microcontroller embedded in the prosthesis form a close loop control and allow the user to produce natural gait patterns [2-3]. However, the function of such a computerized prosthesis is still limited. The primitive prosthesis control is based entirely on mechanical sensing without knowledge of user intent. Users have to "tell" the prostheses their intended activities manually or using body motion, which is cumbersome and does not allow smooth task transitions. The fundamental limitation on all existing prosthetic legs is lack of neural control that would allow the artificial legs to move naturally as if they were the patient's own limb.

This paper presents a novel neural-machine interface (NMI) that makes neural controlled artificial legs possible. The new NMI is a cyber physical system (CPS), in which a complex physical system (i.e. neuromuscular control system of a leg amputee) is monitored and deciphered in real time by a cyber system. It senses neural control

signals from leg amputees, interprets such signals, and makes accurate decisions for prostheses control. The neural signals that our NMI senses and collects from leg amputees are Electromyographic (EMG) signals that represent neuromuscular activity and are effective biological signals for expressing movement intent [4]. EMG signals have been used in many engineering applications, such as EMG-based power-assisted wheelchair [5], biofeedback therapeutic manipulator for lower limb rehabilitation [6], neuro-fuzzy interference system for identifying hand motion commands [7], and neural controlled artificial arms [8-9]. Previous research has shown that EMG was effective and clinically successful for artificial upper limbs [8-9]. However, no EMG-controlled lower limb prosthesis is currently available, and published studies in this area are very limited because of the following technical challenges.

First of all, in human physiological systems, EMG signals recorded from leg muscles during dynamic movements are highly non-stationary. Dynamic signal processing strategies [10] are required for accurate decoding of user intent from such signals. In addition, patients with leg amputations may not have enough EMG recording sites available for neuromuscular information extraction due to the muscle loss [10]. Maximally extracting neural information from such limited signal sources is necessary and challenging.

The second important challenge is that the accuracy in identifying the user's intent for artificial legs is more critical than that for upper limb prostheses. A 90% accuracy rate might be acceptable for control of artificial arms, but it may result in one stumble out of ten steps, which is clearly inadequate for safe use of artificial legs. Achieving high accuracy is further complicated by environmental uncertainty, such as

perspiration, temperature change, and movement between the residual limb and prosthetic socket may cause unexpected sensor failure, influence the recorded EMG signals, and reduce the trustworthiness of the NMI [11]. It is therefore critical to develop a reliable and trustworthy NMI for safe use of prosthetic legs.

The third challenge is the compact and efficient integration of software and hardware in an embedded computer system in order to make the EMG-based NMIs practical and available to patients with leg amputations. Such an embedded system must provide high speed and real time computation of neural deciphering algorithm because any delayed decision-making from the NMI also introduces instability and unsafe use of prostheses. Streaming and storing multiple sensor data, deciphering user intent, and running sensor monitoring algorithms at the same time superimpose a great challenge to the design of an embedded system for the NMI of artificial legs.

To tackle these challenges, a neural interfacing algorithm has been developed that takes EMG inputs from multiple EMG electrodes mounted on a user's lower limb, decodes the user's intended lower limb movements, and monitors sensor behaviors based on trust models. Our EMG pattern recognition (PR) algorithm, together with a post-processing scheme, effectively process non-stationary EMG signals of leg muscles so as to accurately decipher the user's intent. The neural deciphering algorithm consists of two phases: offline training and online testing. To ensure the trustworthiness of NMI in an uncertain environment, a real time trust management (TM) module was designed and implemented to examine the changes of the EMG signals and estimate the trust level of individual sensors. The trust information can be used to reduce the impact of untrustworthy sensors on the system performance.

The new deciphering algorithm was implemented on an embedded hardware architecture as an integrated NMI to be carried by leg amputees. The two key requirements for the hardware architecture were high speed processing of training process and real time processing of the interfacing algorithm. To meet these requirements, the newly designed embedded architecture consists of an embedded microcontroller, a flash memory, and a graphic processing unit (GPU). The embedded microcontroller provided necessary interfaces for AD/DA signal conversion and processing and computation power needed for real time control. The control algorithm was implemented on the bare machine with our own memory and IO managements without using existing OS to avoid any unpredictability and variable delays. The flash memory was used to store training data. EMG PR training process involved intensive signal processing and numerical computations, which needs to be done periodically when the system trust value is low. Such computations can be done efficiently using modern GPUs that provide supercomputing performance with very low cost. New parallel algorithms specifically tailored to the multi-core GPU were developed exploiting memory hierarchy and multithreading of the GPU. Substantial speedups of the GPU for training process were achieved, making the classifier training time tolerable in practice.

A complete prototype has been built implementing all the software and hardware functionalities. The prototype was used to carry out real time testing on human subjects. A male patient with unilateral transfemoral amputations was recruited in our experiments for evaluation of the user intent identification module. The goal of our experiments is to use the newly designed NMI prototype to sense, collect, and decode

neural muscular signals of the human subject. Based on the neural signals, the NMI tries to interpret the subject's intent for sitting and standing, two basic but difficult tasks for patients with transfemoral amputations due to the lack of power from the knee joint. The trust management module was also tested on a male able-bodied subject by introducing motion artifacts during the subject's normal sitting and standing task transitions. The detection rate and false alarm rate for distribution detection was evaluated.

Extensive experiments of our NMI on the human subjects have shown promising results. Among the 30 sitting-to-standing transitions and the 30 standing-to-sitting transitions of the amputee subject, our NMI recognized all the intended transitions correctly with a maximum decision delay of 400ms. Our algorithm can also filter out occasional signal disturbances and motion artifacts with 99.37% detection rate and 0% false alarm rate. The videos of our experiments can be found at http://www.youtube.com/watch?v=H3VrdqXfcm8
and http://www.youtube.com/watch?v=6NwtMOw0YS0.

The paper is organized as follows: The next section presents the system architecture and design of the algorithms and embedded system. Section 1.3 describes the experimental settings for our real time testing of the NMI prototype on the amputee and able-bodied subjects. The results of the study are demonstrated in section 1.4, followed by related work in section 1.5, and a conclusion in section 1.6.

1.2 System Architectures

1.2.1 System Architecture

The architecture of neural-machine interface is shown in Figure 1.1. Multiple channels of EMG signals are the system inputs. EMG signals are preprocessed and segmented by sliding analysis windows. EMG features that characterize individual EMG signals are extracted for each analysis window. The system consists of two major pathways: one path for classifying user movement intent and the other for sensor trust evaluation (the dashed blocks in Figure 1.1). To identify user intent, EMG features of individual channels are concatenated into one feature vector. The goal of pattern recognition is to discriminate among desired classes of limb movement based on the assumption that patterns of EMG features at each location is repeatable for a given motion but different between motions [9]. The output decision stream of EMG pattern classifier is further processed to eliminate erroneous task transitions. In the path for sensor trust evaluation, the behaviors of individual sensors are closely monitored by abnormal detectors. A trust manager evaluates the trust level of each

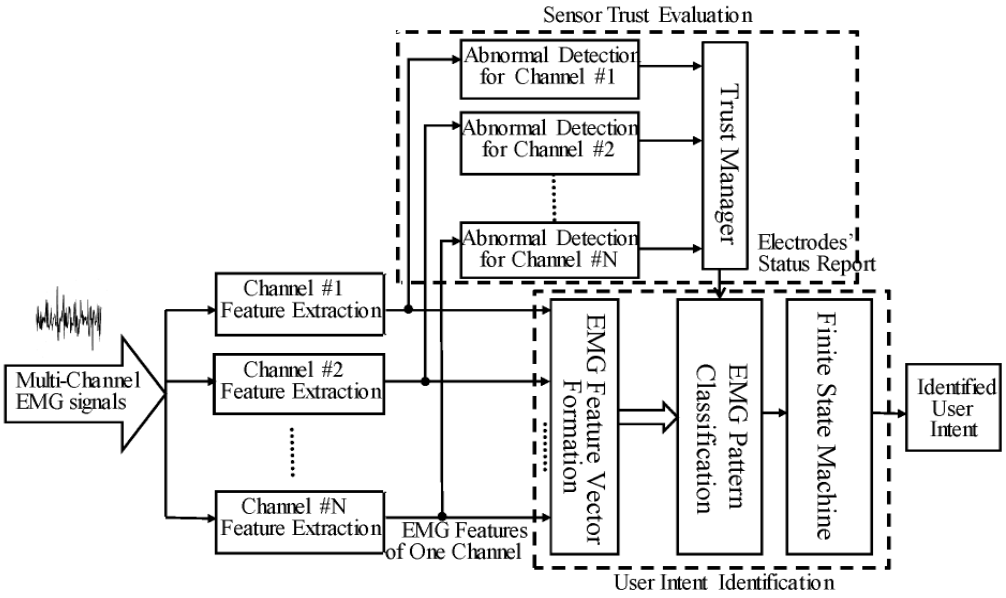

Figure 1.1. Software architecture of EMG-based neural-machine interface for artificial legs.

sensor and then adjusts the operation of the classifier for reliable EMG pattern recognition.

The hardware architecture of the NMI (Figure 1.2) for artificial legs consists of seven components: EMG electrodes, amplifier circuits, analog-to-digital converters (ADCs), flash memory, RAM, GPU and an embedded controller. Multiple channels of EMG signals are collected from different muscles on the patient's residual limb using EMG electrodes. The amplifier circuits are built to make signal polarity, amplitude range, and signal type (differential or single-ended) compatible with the input requirements of ADCs. The outputs of the amplifier circuits are converted to digital format by the ADCs and then stored in a flash memory or a RAM. The embedded hardware works in two modes: training mode and real time testing mode. In the training mode, a large amount of EMG data are collected and stored in the flash memory. These data are then processed to train the EMG pattern classifier. The PR algorithm for the training phase includes complex signal processing and numerical computations, which are done efficiently in a high performance GPU. The parameters

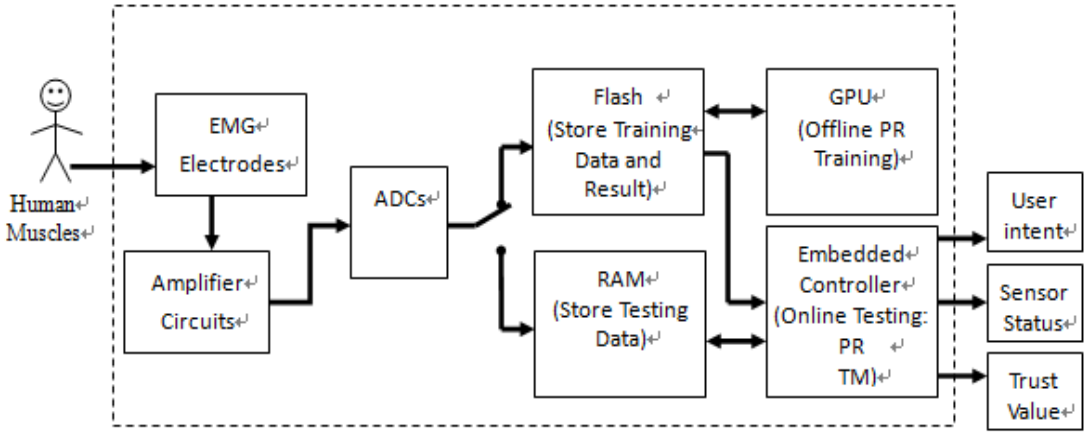

Figure 1.2. Hardware architecture of designed neural-machine interface.

of the trained classifier are stored in the flash memory upon completion of the training phase. The real time testing phase is implemented on the embedded microcontroller, including both the PR algorithm and the TM algorithm. In the real time testing mode, the EMG signals are sampled continuously and stored in the RAM of the embedded controller. The EMG data are then sent to the trained classifier for a decision to identify the user's intended movement and at the same time each EMG sensor is monitored by an abnormal detector. The trust value of each sensor is evaluated by a trust manager.

1.2.2 Identification of User Intent

A dynamic EMG pattern classification strategy and post-processing methods were developed in this study for high decision accuracy.

EMG Signals: EMG signals recorded from gluteal and thigh muscles of residual limb were considered.

EMG Features: Four time-domain (TD) features [12] (the mean absolute value, the number of zero-crossings, the waveform length, and the number of slope sign changes) were selected for real-time operation because of their low computational complexity [9] compared to frequency or time-frequency domain features. The detailed equation and description of these four TD features can be found in [12].

EMG Pattern Classification: Various classification methods, such as linear discriminant analysis (LDA) [12], multilayer perceptron [13], Fuzzy logic [14], and artificial neural network [10, 15], have been applied to EMG PR. The simple LDA classifier was used in this study because of the comparable classification accuracy to more complex classifiers [9, 16-18] and the computation efficiency for real-time

prosthesis control [9].

The idea of discriminant analysis is to classify the observed data to the movement class in which the posteriori probability $P(C_g \mid \bar{f})$ can be maximized. Let $C_g (g \in [1, G])$ denote the movement classes and $\bar{f}$ be the feature vector in one analysis window. The posteriori probability is the probability of class C_g given the observed feature vector $\bar{f}$ and can be expressed as

$$P(C_g \mid \bar{f}) = \frac{P(\bar{f} \mid C_g) P(C_g)}{P(\bar{f})} \tag{1.1}$$

where $P(C_g)$ is the priori possibility, $P(\bar{f} \mid C_g)$ is the likelihood, and $P(\bar{f})$ is the possibility of observed feature vector $\bar{f}$. Given the movement class C_g, the observed feature vectors have a multivariate normal (MVN) distribution. In addition, the priori possibility is assumed to be equivalent for each movement class, and every class shares a common covariance. Hence, the maximization of posteriori possibility in (1.1) becomes

$$\tilde{C}_g = \arg\max_{C_g} \{ \bar{f}^T \Sigma^{-1} \mu_g - \frac{1}{2} \mu_g{}^T \Sigma^{-1} \mu_g \}. \tag{1.2}$$

The following expression,

$$d_{C_g} = \bar{f}^T \Sigma^{-1} \mu_g - \frac{1}{2} \mu_g{}^T \Sigma^{-1} \mu_g, \tag{1.3}$$

is defined as the linear discriminant function, where μ_g is the mean vector and Σ is the common covariance matrix.

During the offline training, μ_g and Σ were estimated by feature vectors calculated from a large amount of training data and were stored in the flash memory.

Let

$$\tilde{\mu}_g = \frac{1}{K_g}\sum_{k=1}^{K_g} \bar{f}_{C_g,k}$$

and

$$\tilde{\Sigma} = \frac{1}{G}\sum_{g=1}^{G}\frac{1}{K_g - 1}(F_g - M_g)(F_g - M_g)^T$$

where K_g is the number of observations in class C_g; $\bar{f}_{C_g,k}$ is the k^{th} observed feature vector in class C_g; F_g is the feature matrix $F_g = [\bar{f}_{C_g,1}, \bar{f}_{C_g,2}, ... \bar{f}_{C_g,k}, ..., \bar{f}_{C_g,K_g}]$; M_g is the mean matrix $M_g = [\tilde{\mu}_g, \tilde{\mu}_g, ... \tilde{\mu}_g]$ that has the same number of columns as in F_g. Then, the parameters in the linear discriminant function (1.3) were known, i.e.

$$\tilde{d}_{C_g} = \bar{f}^T \tilde{\Sigma}^{-1} \tilde{\mu}_g - \frac{1}{2}\tilde{\mu}_g{}^T \tilde{\Sigma}^{-1} \tilde{\mu}_g . \qquad (1.4)$$

In the real time testing, the observed feature $\bar{f}$ derived from each analysis window was fed to the classifier to calculate $\tilde{d}_{C_g}$ in (1.4) for each movement class and was classified into a specific class $\tilde{C}_g$ that satisfied

$$\tilde{C}_g = \arg\max_{C_g}\{\tilde{d}_{C_g}\}, C_g \in \{C_1, C_2, ..., C_G\} .$$

Dynamic Pattern Classification Strategy: When EMG signals are non-stationary, the EMG features across time show large variation within the same task mode, which results in overlaps of features among classes and therefore low accuracy for PR [10]. By assuming that the pattern of non-stationary EMGs has small variation in a short-time window and EMG patterns are repeatable for each defined short-time phase, a

phase-dependent EMG classifier was designed, which was successfully applied to accurately and responsively recognize the user's locomotion modes [10]. For non-locomotion modes such as sitting and standing, the classifier can be built in the movement initiation phase by the same design concept. The structure of such a dynamic design of the classifier can be found elsewhere [10].

Post-processing of Decision Stream: Majority vote was used to eliminate erroneous decisions from the classifier. Majority vote [9] simply removes the decision error by smoothing the decision output. Note that this method can further increase the accuracy of NMI, but may sacrifice the system response time.

1.2.3 Trustworthy Sensor Interface

The NMI for artificial legs must be reliable and trusted by the prosthesis users. The design goals of trustworthy sensor are (1) prompt and accurate detection of disturbances in real time applications, and (2) assessment of reliability of a sensor/system with potential disturbances. To achieve these goals, a trust management module that contains three parts: abnormal detection, trust manager, and decision support was designed.

Abnormal Detection: For each EMG channel, an abnormal detector is applied to detect disturbances occurring in the EMG signal. Disturbances that cause sensor malfunctions can be diverse and unexpected. Among all these disturbances, motion artifacts can cause large damage and are extremely difficult to totally remove. Motion artifacts are also fairly common in both laboratory environment and in real systems. Therefore, in this paper, the focus was on the detection of motion artifacts.

To detect abnormality in EMG signals, a change detector that identifies changes

in the statistics of EMG signals was proposed. During preliminary study, it was found that motion artifacts can lead to changes in two time-domain (TD) features: mean absolute value (increase) and the number of slope sign changes (decrease). Let Fe_{mean} and Fe_{slope} denote these two features, respectively. Positive change in Fe_{mean} and negative change in Fe_{slope} are used as indicators of the presence of motion artifacts. Moreover, since the changes are in two directions, a two-sided change detector, which can detect both positive change and negative change, is required.

Many statistical methods can be used to build the change detector. In this work, the Cumulative Sum (CUSUM) algorithm was chosen because it is reliable for detecting small changes, insensitive to the probabilistic distribution of the underlying signal, and optimal in terms of reducing the detection delay [19]. Particularly, the two-sided CUSUM detector was adopted [20].

$$S_{hi}(i) = \max(0, S_{hi}(i-1) + x_i - \hat{\mu}_0 - k) \tag{1.5}$$

$$S_{lo}(i) = \max(0, S_{lo}(i-1) + \hat{\mu}_0 - k - x_i) \tag{1.6}$$

where x_i represents the i^{th} data sample, $\hat{\mu}_0$ is the mean value of data without changes, and k is CUSUM sensitivity parameter. The smaller the k is, the more sensitive the CUSUM detector is to small changes. In (1.5) and (1.6), S_{hi} and S_{lo} are used for detecting the positive and negative changes, respectively. If S_{hi} (or S_{lo}) exceeds a certain threshold (Th), a positive (or negative) change is detected. The initial values of S_{hi} and S_{lo} were set to 0. In the real time testing, once CUSUM detector detects a change, it will raise an alarm and restart by setting S_{hi} and S_{lo} as 0 in order to detect the next change. By doing so, it can respond sensitively and

promptly to multiple changes in the EMG signal.

The presence of a positive change in Fe_{mean} and a negative change in Fe_{slope} at the same time can serve as the indicator of a motion artifact. Therefore, S_{hi} is applied to detect positive changes in Fe_{mean} and S_{lo} is applied to detect negative changes in Fe_{slope}. When S_{hi} and S_{lo} exceed their corresponding thresholds at the same time, a motion artifact is detected.

In (1.5), x_i denotes the i^{th} sample of Fe_{mean}, and is calculated as mean of the absolute value of EMG signal within the i^{th} window. In (1.6), x_i denotes the i^{th} sample of Fe_{slope}, and is calculated as number of the slope sign changes within the i^{th} window. The value $\hat{\mu}_0$ in (1.5) and (1.6) is computed as the average of x_i before any changes were detected. The sensitivity parameter, k, is set as 0.05, and the threshold Th is set as 0.1 for both (1.5) and (1.6).

Notice that, to promptly respond to disturbances, CUSUM detector restarts for the next round of disturbance detection right after it detects a disturbance. However, there may be a disturbance lasting for some time and CUSUM detector would detect it for more than once. This may lead to an inaccurate trust calculation. To avoid this problem, a post processing scheme is proposed to stabilize the detection result. Disturbances that are very close to each other are combined (i.e. within L continuous windows) as one disturbance. In our real time testing, L is set as 3, which represents 240ms. That is, if the detector is triggered repeatedly within 240ms, we consider this as one disturbance.

Trust manager: After the abnormal detector detects the disturbance in an EMG

signal, the EMG sensor is either permanently damaged or perfectly recoverable. To evaluate the trust level of the sensor, let p_1 denotes the probability that a sensor behaves normally after one disturbance is detected.

Assume all disturbances are independent. The probability that a sensor is still normal after i disturbances, denoted by p_i, is $p_i = p_1^{\,i}$. The trust value is computed from the probability value by the entropy-based trust quantification method [21], as

$$T = \begin{cases} 1 - H(p_i), if\ 0.5 \le p_i \le 1 \\ H(p_i) - 1, if\ 0 \le p_i \le 0.5 \end{cases}$$

where T is the trust value and $H(p_i)$ is the entropy calculated as

$$H(p_i) = -p_i \log_2(p_i) - (1 - p_i)\log_2(1 - p_i). \quad (1.7)$$

Different p_1 values should be set according to the nature of the disturbance. The larger the p_1 value, the less likely the disturbance can damage the sensor. The calculation of trust is extendable to the case that different disturbances are detected for one sensor. For example, if two disturbances, whose p_1 values are 0.8 and 0.9, respectively, are detected for a sensor, the p_i value in (1.7) can be replaced by 0.8×0.9. In this paper, only one type of disturbance (i.e. motion artifact) was tested. The p_1 value for motion artifact is set as 0.9.

Decision Making and Report: The trust information is provided to the user intent identification (UII) module to assist trust-based decisions. There are two levels of decisions.

1) Sensor level: When the sensor's trust value drops below a threshold, this sensor is considered as damaged, and its reading is removed from the UII module. The

classifier needs to be re-trained without the damaged sensor.

2) System level: After removing the damaged sensors, the system trust can be calculated by the summation of trust values of the remaining sensors. If the system trust is lower than a threshold, this entire UII model is not trustworthy, and actions for system recovery must be taken. One possible action is to re-train the classifier. Another possible action is to instruct the patient to manually examine the artificial leg system.

1.2.4 Hardware Design

Technical challenges in hardware design are twofold. First of all, in order to increase the decision accuracy, frequent training computations may often be required, especially in uncertain environment, where the appearance of disturbances can be unpredictable and frequent. A training computation needs to be done not only whenever the user puts on the prosthesis but also whenever the system trust level goes below the predetermined threshold. Training data need to be recollected in these two cases. In addition, when a sensor's trust value drops below a threshold, the classifier also needs to be re-trained using existing training data in the flash memory such that the classifier can make decisions based on the remaining undisturbed sensors. The training algorithms require intensive numerical computations that take a significantly long time, in the range of a few minutes to hours on a general purpose computer system [22]. It is very important to substantially speed up this training computation to make the training time of our NMI tolerable and practical. The second challenge is the real time processing of decision making in order to have smooth control of the artificial legs. Such real time processing includes signal sampling, AD/DA conversion,

storing digital information in memory, executing PR algorithms, periodical trust management, and decision outputs. To meet these technical challenges, a new hardware design incorporating a multi-core GPU and an embedded system with a built-in flash memory was presented.

High performance and low cost multi-core GPUs [23] have traditionally been thought of as commodity chips to drive consumer video games. However, the push for realism in such games along with the rapid development of semiconductor technologies has made GPUs capable of supercomputing performance for many applications at very low cost. There are many low-end to medium GPU controller cards available on the market for under $50. However they deliver extraordinary computation power in the range of several hundreds of GFLOPS. Besides high performance and low cost, there has also been a technology drive for reliable and low power GPUs alongside FPGAs and CPUs for embedded applications such as military systems. For example, an embedded system using the ATI Radeon HD 3650 GPU draws very little power but delivers performance levels of hundreds of GFLOPS. The next-generation mobile GPUs are expected to nearly double this performance with a similar power envelope. Our NMI makes the first attempt to exploit such high speed and low cost GPU for the purpose of speeding up complex PR training computations. Our design for the training of the classifier used a NVIDIA 9500GT graphic card that has four multiprocessors with 32 cores working at the clock rate of 1.4 GHz. Each multiprocessor supports 768 active threads giving rise to a total of 3072 threads that can execute in parallel. These threads are managed in blocks. The maximum number of threads per block is 512. The size of the global memory is 1 GB with bandwidth of

25.6 GB/s. 64 KB of the global memory is read-only constant memory. The threads in each block have 16 KB shared memory which is much faster than the global memory because it is cached. In this study, this GPU card was connected using the x16 PCI Express bus. Whenever the training computation was triggered, the GPU was called in to perform the training process and store the parameters of trained classifier in the flash memory to be used for real time decision-making.

The second part of the hardware design is based on Freescale's MPC5566 132 MHz 32 bits microcontroller unit (MCU) with the Power Architecture as shown in Figure 1.3. The MCU has 40 channels of ADCs with up to 12 bit resolution and two levels of memory hierarchy. The fastest memory is 32KB unified cache. The lower level memories include 128KB SRAM and 3MB flash memory. The default system clock of the MCU is 12 MHz. The frequency modulated phase locked loop (FMPLL) generates high speed system clocks of 128 MHz from an 8 MHz crystal oscillator. The

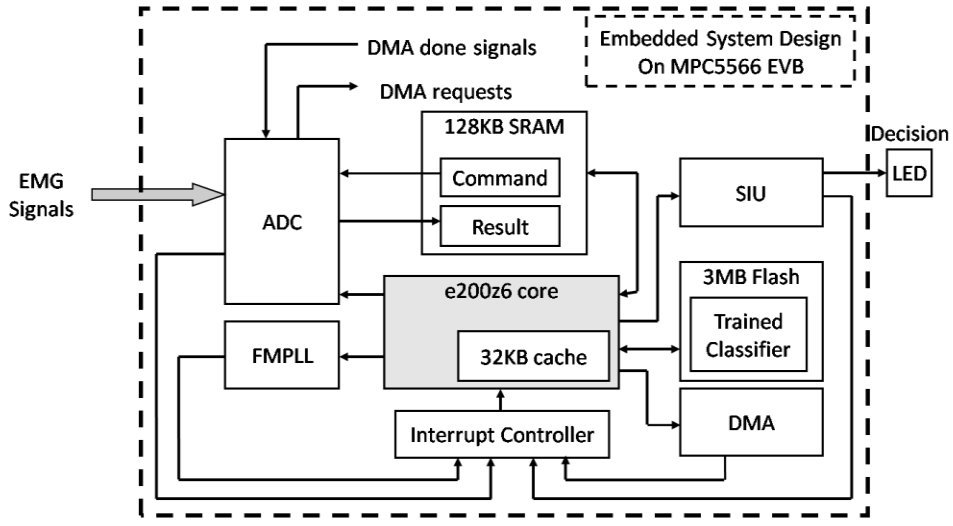

Figure 1.3. Block diagram of embedded system design on MPC5566 EVB for real-time testing. MPC5566: device modules; ADC: analog-to-digital converter; FMPLL: frequency modulated phase-locked loop; SRAM: internal static RAM; SIU: system integration unit; DMA: direct memory access.

direct memory access (DMA) engine transfers the commands and data between SRAM and ADC without direct involvement of the CPU. Minimizing the intervention from CPU is important for achieving optimal system response. The device system integration unit (SIU) configures and initializes the control of general-purpose I/Os (GPIOs). The real-time results of the embedded system, including the identified user intent, individual sensor status and trust value, are sent to the GPIO pins and displayed by multiple LEDs on MPC5566 EVB.

1.3 Experiments and Prototype

1.3.1 Evaluation of Designed Algorithm

Assigned Tasks: To prove the design concept, the NMI system was designed to decipher the task transitions between sitting and standing. These tasks are the basic activities of daily living but difficult for patients with transfemoral amputations due to the lack of knee power. During the transition phase, EMG signals are non-stationary. The classifier was designed in the short transition phase. Although it is possible to activate the knee joint directly based on the magnitude of one EMG signal or force data recorded from the prosthetic pylon, unintentional movements of the residual limb in the sitting or standing position may accidently activate the knee, which in turn may cause a fall in leg amputees. Hence, intuitive activation of a powered artificial knee joint for mode transitions requires accurate decoding of EMG signals for identifying the user's intent from the brain.

Data Collection: This study was conducted with Institutional Review Board (IRB) approval at the University of Rhode Island and informed consent of subjects. For the real time evaluation of the designed pattern recognition algorithm, one male

patient with a unilateral transfemoral amputation was recruited. To evaluate the sensor trust algorithm, one male able-bodied subject, free from orthopedic or neurological pathologies, was recruited. Seven surface EMG electrodes (MA-420-002, Motion Lab System Inc., Baton Rouge, LA) were used to record signals from gluteal and thigh muscles in one side of both subjects. The EMG electrodes contained a pre-amplifier which band-pass filtered the EMG signals between 10 Hz and 3,500 Hz with a pass-band gain of 20. For the able-bodied subject, the gluteal and thigh muscles on the dominant leg were monitored. After the skin was shaved and cleaned with alcohol pads, the EMG electrodes were placed on the anatomical locations described in [24]. For the amputee subject, the muscles surrounding the residual limb and the ipsilateral gluteral muscles were monitored. The subject was instructed to perform hip movements and to imagine and execute knee flexion and extension. We placed EMG electrodes at the locations, where strong EMG signals can be recorded. EMG electrodes were embedded into a customized gel-liner system (Ohio Willow Wood, US) for reliable electrode-skin contact. A ground electrode was placed near the anterior iliac spine for both able-bodied and amputee subjects. An MA-300 system (Motion Lab System Inc., Baton Rouge, LA) collected 7 channels of EMG data. The cut-off frequency of the anti-aliasing filter was 500 Hz for EMG channels. All the signals were digitally sampled at a rate of 1000 Hz and synchronized.

The states of sitting and standing were indicated by a pressure measuring mat. The sensors were attached to the gluteal region of the subject. During the weight bearing standing, the recordings of the pressure sensors were zero; during the non-weight bearing sitting, the sensors gave non-zero readings.

Experiment Protocol: To evaluate the pattern recognition algorithm, before the real-time system testing, a training session was required in order to collect the training data for the classifier. During the training session, the subject was instructed to perform four tasks (sitting, sit-to-stand, standing, and stand-to-sit) on a chair (50 cm high). For sitting or standing task, the subject was required to keep the position for at least 10 sec. In the sitting or standing position, the subject was allowed to move the legs and shift the body weight. For two types of transitions, the subject performed the transitions without any assistance at least 5 times. During the real-time system evaluation testing, the subject was asked to sit and stand continuously. A total of 5 trials were conducted. In each trial, the subject was required to sit and stand at least five times, respectively. Rest periods were allowed between trials in order to avoid fatigue.

To evaluate the sensor trust algorithm, 13 trials of real-time disturbance detection testing were performed on the able-bodied subject. In each trial, motion artifacts were introduced randomly on one EMG electrode in each task phase for four times. To add motion artifacts, the experimenter tapped an EMG electrode with roughly same strength. Motion artifacts were introduced 159 times in the entire experiment.

Real-time Evaluation of EMG Pattern Recognition: Four classes during the movement initiation phase were considered: sitting, sit-to-stand transition, standing, and stand-to-sit transition. Note that the classes of sitting and standing were not stationary because the subject was instructed to move the legs and shift the body weight in these positions. The output of the classifier was further combined into two classes (class 1: sitting and stand-to-sit transition; class 2: standing and sit-to-stand

transition). Four TD features defined in [12] and LDA-based classifier were used. Overlapped analysis windows were used in order to achieve prompt system response. For the real-time algorithm evaluation, 140ms window length and 80ms window increment were chosen. Two indicators were used to evaluate the real-time performance of EMG pattern classifier: classification accuracy and classification response time. Two types of classification response time were defined: the time delay (RT1) between the moment that the classification decision switched from sitting (0) and standing (1) and the moment that the gluteal region pressure changed from non-zero value (non-weight bearing sitting) to zero value (weight-bearing standing); the time delay (RT2) between the moment that the classification decision switched from standing (1) to sitting (0) and the moment that the gluteal region pressure changed from zero value (weight-bearing standing) to non-zero value (non-weight bearing sitting).

Real-Time Evaluation of Abnormal Detection and Trust Management: EMG electrodes recorded EMG signals under the task transitions, unintentional leg movements, and sensor disturbances. There were two different states: (1) normal movements (N), including unintentional leg movements and transitions between sitting and standing, the total number of which were 364, and (2) disturbances (D), the total number of which were 159. The detectors detected two types of results: normal (N) or disturbance (D).

For the data sets with motion artifacts, the data in each trial were divided into analysis windows. A state (N or D) was assigned to each window. There were four detection results: (1) Hit (H): Truth = ‘D’, Detection = ‘D’; (2) False Alarm (F): Truth

= ‘N’, Detection = ‘D’; (3) Miss Detection (M): Truth = ‘D’, Detection = ‘N’; and (4) Correct no detection (Z): Truth = ‘N’, Detection = ‘N’. The performance of designed detector were evaluated by

Probability of detection: $PD = \frac{H}{H + M}$

Probability of false alarm: $PFA = \frac{F}{F + Z}$

The trust value of sensors will also be shown.

1.3.2 Algorithm Implementation on NMI Hardware System

The offline PR training algorithm, the real time PR testing algorithm, and the real time TM algorithm were all implemented on the NMI hardware described in the previous section. The window length and the window increment were set to 140ms and 80ms, respectively. This is because the computation speed of MPC5566 is limited. It takes approximate 80ms to compute the EMG PR algorithm and to run the abnormal detection/trust evaluation algorithm on the data collected in a 140ms window. Therefore, the window increment should be no less than 80ms. It was observed in our experiments that enlarging the window length exceeding 120ms does not affect the classification performance [10] but increases the decision-making time , which causes delayed system response.

A parallel algorithm specially tailored to the GPU architecture for the computation intensive part of the PR training algorithm was designed using CUDA: Compute Unified Device Architecture, which is a parallel computing engine developed by NVIDIA. At the time of this experiment, our GPU was not directly connected to the embedded MCU. Rather, NVIDIA 9500GT graphics card plugged

into the PCI-Express slot of the PC server was used to do the training computation. The training results were then manually loaded into the flash memory of the embedded system board for real time testing. The GPU took inputs from 7 EMG channels, each of which had about 10,000 data points. The EMG data were segmented into analysis windows with 140ms in length. As a result, each window contained a 140×7 matrix. The training algorithm first extracted 4 TD features from each channel, producing a 28×1 feature vector for each window. Our parallel algorithm on the CUDA spawned 7 threads for each window resulting in totally 2,800 threads for 400 windows. All these threads were executed in parallel on the GPU to speed up the process. The resultant features were stored in a $28 \times W$ matrix, where W is the number of windows. The algorithm then set up K thread blocks, where K is the number of observed motions of the user. Each one of the K thread blocks had 28×14 threads, and a total of $K \times 28 \times 14$ threads could execute simultaneously in parallel on the GPU architecture.

To demonstrate the speedup provided by our parallel implementation on the GPU, an experiment that compared the computation times of our training algorithm on both the GPU system and the fully equipped 3 GHz Pentium 4 PC server was conducted. (The results will be shown in Section 14.3.)

The real time testing algorithm was implemented on Freescale's MPC5566 evaluation board, integrating both the PR algorithm for user intent identification and the TM algorithm for sensor trust evaluation. The parameters of the trained PR classifier, a 28×4 matrix and a 1×4 matrix, calculated during the training phase by GPU were stored in the built-in flash memory on the MPC5566 EVB in advance. The ADCs sampled raw EMG data of 7 channels at the sampling rate of 1000 Hz

continuously. Same as in the training phase, the EMG data were divided into windows of length 140ms and increment 80ms. In every analysis window, 4 TD features were extracted for each individual channel. During the user intent identification process, a 28×1 feature vector was derived from each window and then fed to the trained classifier. After the EMG pattern classification, one movement class out of four was identified. The result was post-processed by the majority vote algorithm to produce a final decision – sitting or standing. During the sensor trust evaluation process, each EMG sensor was monitored by an individual abnormal detector. Only two of the four TD features (the mean absolute value and the number of slope sign changes) were used to detect motion artifacts (algorithm details in Section 1.2.3). Each abnormal detector monitored the changes of these two TD features to produce a status output for its corresponding sensor: normal or disturbed. A trust level manager then evaluated the trust level of individual sensor based on accumulated disturbance information.

In the real time embedded system design, to ensure smooth control of the artificial legs, precise timing control and efficient memory management are two challenges due to the speed and memory limitations of the embedded controller. We developed our own hardware management mechanism on the bare machine of the MPC5566 EVB without depending on any real time OS to avoid unpredictability and delay variations. A circular buffer was designed to allow simultaneous data sampling and decision making. The circular buffer consisted of three memory blocks *B1*, *B2* and *B3* that were used to store the ADC sampling data. Each block stored the data sampled in one window increment (80 samples in this experiment). An additional memory block, *B4*, was used as a temporary storage during the computation of PR algorithm and TM

algorithm.

Figure 1.4 shows the timing diagram of the control algorithm during the real time testing process. In Figure 1.4, Δt equals the window increment, t_{PR} is the execution time of PR algorithm, and t_{TM} is the execution time of TM algorithm. Two conditions need to be satisfied to ensure the smooth control of decision making without delay: (1) $t_{TM}+t_{PR} \le \Delta t$ and (2) $t_w \le 2\Delta t$, where t_w is the window length. At point t_0, the ADCs begin to sample EMG signals continuously and the digital data are stored in *B1*. From point t_1, *B1* is filled up and the in-coming data are stored in *B2*. At t_2, the data for the first window *W1* are available (stored in *B1* and *B2*), and an interrupt request is generated to notify the CPU that the data is ready for computation and trigger the computation to start. At the same time, new data keep coming in to be stored in *B3*. After the time interval $t_{TM}+t_{PR}$, at point t_3, the PR computation and the sensor trust

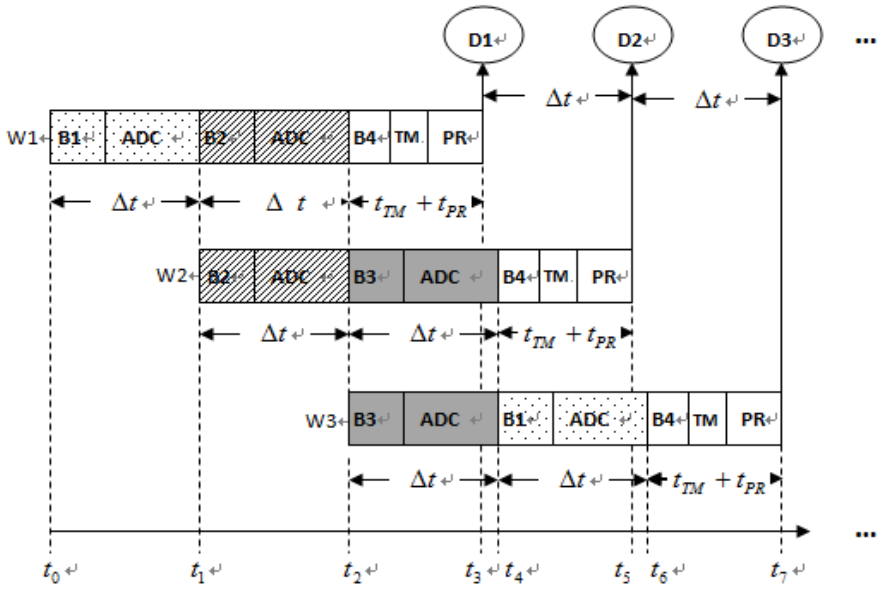

Figure 1.4. Timing control of real-time decision making.

computation of *W1* complete. The first decision *D1* is made, identifying the user's intent of window *W1* whether to sit or stand, and also reporting the status and the trust value of each sensor. At time t_4, *B3* is filled up and data for *W2* as a result of sliding 80 ms are ready for the computation again, using data partly in B2 and partly in B3. At this time, *B1* is no longer in use so it can be replaced by new sampling data. At time t_5, the decision *D2* of window *W2* is made. At time t_6, data for *W3* (stored in *B3* and *B1*) are available, the algorithm computation for *W3* begins. At time t_7, *D3* is done and *B2* can be reused.

1.3.3 Real-Time Testing of the NMI Prototype

Using the NMI prototype described above, real time tests were carried out as described in Section 1.3.1. At the time of this experiment, our trust model focused on abnormal detection and the trust was evaluated at the sensor level. The communication between the trust manager and the classifier was not fully considered. Therefore, to better evaluate our system performance, a two-phase experiment was set up to evaluate the performance of pattern recognition and that of sensor trust management separately. For both phases, the subjects performed transitions between sitting and standing continuously. During *the phase of PR evaluation*, there was no motion artifact manually added. However, the subject's unintentional movements and the movements between the residual limb and prosthetic socket were still a factor. The movement decisions made by the classification system were displayed on a LED light and a computer monitor in real time. In our experiment, a 5-window majority vote was applied to the decision stream to further eliminate the classification errors. During *the phase of sensor trust evaluation*, motion artifacts were manually introduced by

randomly tapping an EMG electrode with roughly equal strength. The sensor status and the sensor trust value were monitored and displayed on a computer monitor. The user intent classification results were ignored during this phase.

1.4 Results and Discussions

1.4.1 Real-Time Performance of Pattern Recognition

During the continuous real-time testing (more than 30 times sit-to-stand transitions and 30 times stand-to-sit transitions), all of the transitions between sitting and standing were accurately recognized. Although the subject moved the legs during the sitting position and shifted the body weight in the standing position, no classification error was observed.

The system classification response time (RT1 and RT2) was calculated by using the pressure data under the gluteal region and shown in Table 1.1.

The real-time performance of the designed NMI prototype in one representative trial is shown in Figure 1.5. Due to a 5-window majority vote method applied, around 400ms decision delay for the sit-to-stand transitions were observed in Figure 1.5, comparing to the falling edges of pressure data. It can be clearly seen that the majority

Table 1.1. System classification response time

RT1	RT2
+(364±38) ms	-(875±27) ms

Note: '+' represents the classification decision was made after the event (non-weight bearing sitting to weight-bearing standing); '-' means the classification decision was made before the event (weight-bearing standing to non-weight bearing sitting)

vote post-processing method significantly improved system accuracy but sacrificed system response time. The video of real-time system performance can be found at http://www.youtube.com/watch?v=H3VrdqXfcm8.

Compared to the real-time testing results on one able-bodied subject (upper two photos in Figure 1.6) in our experiments [25], a similarly high classification accuracy and reasonable system response time were achieved on the patient with transfemoral amputation (lower two photos in Figure 1.6). The promising real-time performance of our designed NMI prototype demonstrates a great potential to allow the amputee patients to intuitively and efficiently control the prosthetic legs.

1.4.2 Real-Time Performance of Sensor Trust Algorithm

Figure 1.7 shows the performance of the designed trust management method. There are three subfigures. The upper figure shows the EMG signal disturbed by

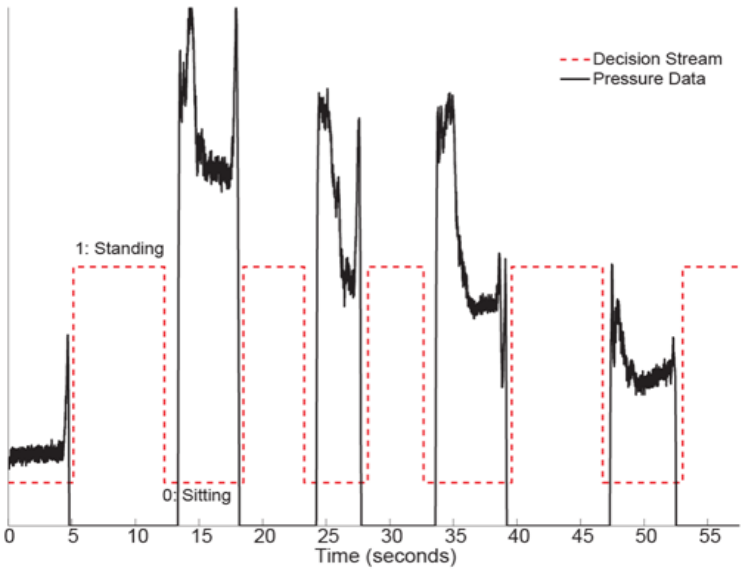

Figure 1.5. Real-time performance of the designed NMI system. The decision stream (0: sitting, 1: standing) is aligned with the pressure data (black solid line) measured under the gluteal region of the subject.

Figure 1.6. Real-time testing of the designed NMI prototype on human subjects. Lower two photos show the experiments on one patient with transfemoral amputation. Upper two photos show the experiments on an able-bodied subject.

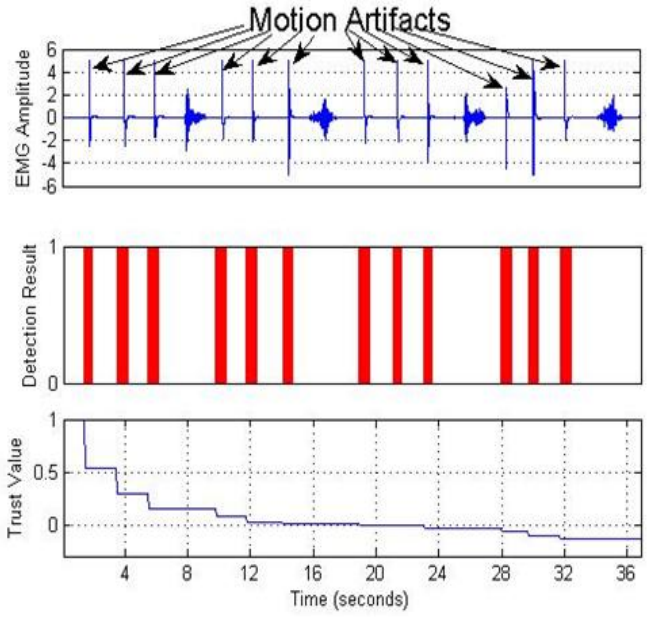

Figure 1.7. Real-time performance demonstration of the abnormal detector under motion artifacts. The representative EMG signals(upper panel), the detection results of CUSUM (middle panel), and the trust value (lower panel) are demonstrated.

motion artifacts. The middle one shows the CUSUM detection results, the bar represents the period that a motion artifact was detected. As seen in the figure,

CUSUM detector was sensitive to motion artifacts, but insensitive to the muscle activity due to the normal leg movements. Additionally, the CUSUM had very small detection delay. The bars were always present immediately after a motion artifact. The lower figure shows the corresponding trust value. The trust value for motion artifacts gradually reduced when consistent disturbances were detected. In the future work, other methods for trust value calculation will be explored. For instance, for sensors with non-perfect trust values, it can be checked whether their future readings are consistent with other sensors that have high trust values. By doing so, the sensors that experienced an occasional disturbance and were not damaged can gradually regain the trust. Furthermore, the performance of CUSUM detector was evaluated by calculating its detection rate and false alarm rate. During the real time testing experiments, **CUSUM detector achieved 99.37% detection rate and 0% false alarm rate.** The undetected disturbances are disturbances with either small amplitude or short duration, so that a small number of such disturbances were not expected to affect the NMI decision significantly. The video of our experiment can be found at http://www.youtube.com/watch?v=6NwtMOw0YS0.

The designed CUSUM detector is accurate and prompt. The limitations of the current study are that only one electrode was disturbed and the trust manager evaluated the trust only at the sensor level. In the next design phase, we will (1) consider the situation with multiple sensor failures, (2) enable the communication between the trust manager and the classifier, and (3) evaluate the system-level trust of the entire NMI.

1.4.3 Performance of CPU vs. GPU for Training Procedure

Table 1.2 shows the measured speedup of our parallel algorithm on the NVIDIA GPU over the PC server for different window sizes. It is clear from this table that our parallel implementation on the GPU gives over an order of magnitude speedup over the PC server. This order of magnitude speedup is practically significant. Consider the case where the training time took half hour on a PC server [22]. The same training algorithm takes less than a minute using our new parallel algorithm on the GPU. From an amputee user point of view, training for less than a minute for the purpose of accurate and smooth neural control of the artificial leg is fairly manageable as compared to training for half an hour every time training is necessary. Furthermore, the speedup increases as the number of windows increases (Table 2). As a result, parallel computation of the training algorithm on GPU helps greatly in the NMI design since the larger the number of windows, the higher its decision accuracy will be [25].

1.4.4 Discussions

While our experiments on two subjects, one able-bodied subject and the other transemoral amputee subject have demonstrated promising results of the new NMI prototype, system performance may vary among different subjects due to the inter-subject variations. One of our future research tasks is to recruit more amputee subjects with diverse age and gender groups to evaluate the performance of the new NMI

In this presented study, only two simple tasks (sitting and standing) were tested. To allow the prosthesis to perform more functions, more locomotion modes during the

Table 1.2. Speedups of the GPU parallel training algorithm over the 3GHz PC server.

Window size	100	200	400	600	800
Speedup	22.98	29.50	35.94	37.16	39.21

ambulation such as level ground walking, stair ascent/descent, and ramp ascent/descent should be considered in our future work. A phase-dependent strategy, which was proposed in our previous study [10], may be required for the designed neural-machine interface structure to handle the walking dynamics.

Among all types of disturbances, motion artifact occurs most frequently in practice and therefore is the main disturbance studied in this paper. Besides motion artifact, there are also some other disturbances which should be considered in our future work, such as baseline noise amplitude change, signal saturation, sensor loss of contact, and etc. To handle these diverse disturbances, we may need to extend the work by (1) applying the abnormal detection on more EMG features and (2) modifying the trust manager so that the trust value is determined not only by how many times disturbances occur but also by more complex factors, such as disturbance type, duration, severe level and etc.

1.5 Related Work

Real-time EMG pattern recognition has been designed to increase the information extracted from EMG signals and improve the dexterity of myoelectric control for upper limb prosthetics [9, 26]. However, no EMG-controlled lower-limb prostheses are currently available. Recently, the need for neural control of prosthetic legs has brought the idea of EMG-based control back to attention. Two previous studies have attempted to use EMG signals to identify locomotion modes for prosthetic leg control [10, 27]. Jin et al. [27] used features extracted from EMG signals from a complete stride cycle. Using such features, the algorithm results in a time delay of one stride cycle in real-time. In practical application, this is inadequate for safe prosthesis use.

Our previous study designed a phase-dependent EMG pattern recognition method [10], which is a dynamic classifier over time. The result indicated over 90% classification accuracy, which can be applied for real time NMI. While both studies demonstrated that EMG information recorded from transfemoral amputees is sufficient for accurate identification of user intent, there has been no experimental study on design and implementation of embedded system to realize the NMI for reliable and real time control of prosthesis.

Reliable EMG pattern recognition system for artificial legs has been developed in our previous study [11]. It can enhance the system performance when sudden disturbances were applied to multiple sensors. In the previous work, however, the disturbances were generated through simulations and the algorithms were only tested offline [25]. The proposed algorithms in this paper, which were very different from the previous approaches, focused on real-time design with low detection latency, and were implemented and tested in a real-time embedded system.

There has been extensive research in using GPUs for general purpose computing (GPGPU) to obtain exceptional computation performance for many data parallel applications [23, 28-30]. A good summary of GPGPU can be found in [23, 31-32]. Our prior study made the first attempt to use GPU in EMG-controlled artificial legs and other medical applications [22]. Our results on individual computation components on EMG signal pattern recognition showed good speedups of GPU over CPU for various window sizes. The focus of the work reported in [22] was on parallel implementations of individual algorithms on GPU whereas this paper makes the first attempt to integrate the entire system for neural-machine interfacing (i.e. a CPS) for

real time control of artificial legs. Our prior works [22] report offline analysis, while the work presented in this paper implements online decoding method for real-time testing. To the best knowledge of the authors, there has been no existing study on implementing the entire training algorithm on GPU for different numbers of windows and integrating the training algorithm together with real time testing on the same subject.

1.6 Conclusions

A new EMG-based neural-machine interface (NMI) for artificial legs was developed and implemented on an embedded system for real time operation. The NMI represents a typical cyber-physical system that tightly integrated cyber and physical systems to achieve high accuracy, reliability, and real-time operation. This cyber-physical system consists of (1) an EMG pattern classifier for decoding the user's intended lower limb movements and (2) a trust management mechanism for handling unexpected sensor failures and signal disturbances. The software was then embedded in a newly designed hardware platform based on an embedded microcontroller and a GPU to form a complete NMI for real time testing. To prove our design concepts, a working prototype was built to conduct experiments on a human subject with transfemoral leg amputations and an able-bodied subject to identify their intent for sitting and standing. We also tested our trust management model on an able-bodied human subject by adding motion artifacts. The results showed high system accuracy, reliability and reasonable time response for real time operation. Our NMI design has a great potential to allow leg amputees to intuitively and efficiently control prosthetic legs, which in turn will improve the function of prosthetic legs and the quality of life

for patients with leg amputations. Our future work includes the consideration of other movement tasks such as walking on different terrains, communications between trust models and user intent identification models, and exploring online training algorithms.

List of References

[1] B. Potter and C. Scoville, "Amputation is not isolated: an overview of the us army amputee patient care program and associated amputee injuries," *Journal of the American Academy of Orthopaedic Surgeons,* vol. 14, p. S188, 2006.

[2] H. Herr and A. Wilkenfeld, "User-adaptive control of a magnetorheological prosthetic knee," *Industrial Robot: An International Journal,* vol. 30, pp. 42-55, 2003.

[3] R. Psonak, "Transfemoral prosthetics," *Orthotics and Prosthetics in Rehabilitation*.

[4] J. V. Basmajian and C. J. De Luca, *Muscles alive : their functions revealed by electromyography*, 5th ed. Baltimore: Williams & Wilkins, 1985.

[5] Y. Oonishi, S. Oh and Y. Hori, "A New Control Method for Power-Assisted Wheelchair Based on the Surface Myoelectric Signal," *IEEE Transactions on Industrial Electronics,* vol. 57, pp. 3191-3196, 2010.

[6] S. Komada, Y. Hashimoto, N. Okuyama, T. Hisada and J. Hirai, "Development of a biofeedback therapeutic-exercise-supporting manipulator," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 3914-3920, 2009.

[7] M. Khezri and M. Jahed, "A Neuro-Fuzzy Inference System for SEMG Based Identification of Hand Motion Commands," *IEEE Transactions on Industrial Electronics,* pp. 1-1.

[8] P. Parker and R. Scott, "Myoelectric control of prostheses," *Critical reviews in biomedical engineering*, vol. 13, p. 283, 1986.

[9] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans Biomed Eng,* vol. 50, pp. 848-54, Jul 2003.

[10] H. Huang, T. A. Kuiken and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Trans Biomed Eng,* vol. 56, pp. 65-73, Jan 2009.

[11] H. Huang, F. Zhang, Y. L. Sun and H. He, "Design of a robust EMG sensing interface for pattern classification," *J Neural Eng,* vol. 7, p. 056005, Oct 2010.

[12] B. Hudgins, P. Parker and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Trans Biomed Eng,* vol. 40, pp. 82-94, Jan 1993.

[13] N. F. Guler and S. Kocer, "Classification of EMG signals using PCA and FFT," *J Med Syst,* vol. 29, pp. 241-50, Jun 2005.

[14] A. B. Ajiboye and R. F. Weir, "A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control," *IEEE Trans Neural Syst Rehabil Eng,* vol. 13, pp. 280-91, Sep 2005.

[15] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Industrial Electronics Magazine,* vol. 3, pp. 56-63, 2009.

[16] Y. Huang, K. B. Englehart, B. Hudgins and A. D. Chan, "A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses," *IEEE Trans Biomed Eng,* vol. 52, pp. 1801-11, Nov 2005.

[17] K. Englehart, B. Hudgins, P. A. Parker and M. Stevenson, "Classification of the myoelectric signal using time-frequency based representations," *Med Eng Phys,* vol. 21, pp. 431-8, Jul-Sep 1999.

[18] L. J. Hargrove, K. Englehart and B. Hudgins, "A comparison of surface and intramuscular myoelectric signal classification," *IEEE Trans Biomed Eng,* vol. 54, pp. 847-53, May 2007.

[19] T. Philips, E. Yashchin and D. Stein, "Using Statistical Process Control to Monitor Active Managers," *Journal of Portfolio Management* vol. 30, pp. 186-91, 2003.

[20] E. S. Page, "Continuous Inspection Scheme," *Biometrika* vol. 41, pp. 100-15, 1954.

[21] Y. Sun, W. Yu, Z. Han and R. Liu, "Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks," *IEEE JSAC special issue on security in wireless ad hoc networks,* vol. 24, 2006.

[22] W. Xiao, H. Huang, Y. Sun and Q. Yang, "Promise of embedded system with GPU in artificial leg control: Enabling time-frequency feature extraction from electromyography," *Conf Proc IEEE Eng Med Biol Soc,* vol. 1, pp. 6926-9, 2009.

[23] D. J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn and T. J. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware," *Computer Graphics Forum,* vol. 26, pp. 80-113, 2007.

[24] A. Perotto, E. F. Delagi, J. Iazzetti and D. Morrison, *Anatomical Guide For The Electromyographer: The Limbs And Trunk*: Charles C Thomas Publisher Ltd, 2005.

[25] H. Huang, Y. Sun, Q. Yang, F. Zhang, X. Zhang, Y. Liu, J. Ren and F. Sierra, "Integrating neuromuscular and cyber systems for neural control of artificial legs," *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, 2010, pp. 129-138.

[26] T. A. Kuiken, G. Li, B. A. Lock, R. D. Lipschutz, L. A. Miller, K. A. Stubblefield and K. B. Englehart, "Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms," *Jama*, vol. 301, pp. 619-28, Feb 11 2009.

[27] D. Jin, J. Yang, R. Zhang, R. Wang and J. Zhang, "Terrain Identification for Prosthetic Knees Based on Electromyographic Signal Features*," *Tsinghua Science & Technology*, vol. 11, pp. 74-79, 2006.

[28] A. Bayoumi, M. Chu, Y. Hanafy, P. Harrell and G. Refai-Ahmed, "Scientific and engineering computing using ati stream technology," *Computing in Science & Engineering*, pp. 92-97, 2009.

[29] J. Montrym and H. Moreton, "The geforce 6800," *Micro, IEEE*, vol. 25, pp. 41-51, 2005.

[30] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang and V. Volkov, "Parallel computing experiences with CUDA," *Micro, IEEE*, vol. 28, pp. 13-27, 2008.

[31] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, pp. 879-899, 2008.

[32] J. Nickolls and W. J. Dally, "The GPU computing era," *Micro, IEEE*, vol. 30, pp. 56-69, 2010.

MANUSCRIPT 2

Design and Implementation of a Special Purpose Embedded System for Neural-Machine Interface

by

[1]Xiaorong Zhang, He Huang, and Qing Yang

is published in *the proceeding of the 28th IEEE International Conference on Computer Design (ICCD'10)*, Amsterdam, Netherland, 2010. p. 166-172.

[1] Xiaorong Zhang, He Huang, and Qing Yang are with Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, 02881, Email {zxiaorong, huang, qyang}@ele.uri.edu.

Abstract

Our previous study has shown the potential of using a computer system to accurately decode electromyographic (EMG) signals for neural controlled artificial legs. Because of computation complexity of the training algorithm coupled with real time requirement of controlling artificial legs, traditional embedded systems generally cannot be directly applied to the system. This paper presents a new design of an FPGA-based neural-machine interface for artificial legs. Both the training algorithm and the real time controlling algorithm are implemented on an FPGA. A soft processor built on the FPGA is used to manage hardware components and direct data flows. The implementation and evaluation of this design are based on Altera Stratix II GX EP2SGX90 FPGA device on a PCI Express development board. Our performance evaluations indicate that a speedup of around 280X can be achieved over our previous software implementation with no sacrifice of computation accuracy. The results demonstrate the feasibility of a self-contained, low power, and high performance real-time neural-machine interface for artificial legs.

2.1 Introduction

The technology of integrating human neuromuscular system and computer system to control artificial limbs has advanced rapidly in recent years. The key to the success of this advancement is the neural-machine interface (NMI) that senses neuromuscular control signals, interprets such signals, and makes decisions to identify user's movement intent for prostheses control. Electromyographic (EMG) signals are effective muscle electrical signals for expressing movement intent for neural control of artificial limbs and can be acquired from electrodes mounted directly on the skin [1].

While EMG-based NMIs have been proposed and clinically tested for artificial arms [2-3], no EMG-controlled prosthetic legs are available because of two main reasons.

(1) Due to the highly non-stationary characteristics of leg EMG signals, accurate decoding of user intent from such signals requires dynamic signal processing strategies [4]. Furthermore, the accuracy in identifying user's intended lower limb movement is more critical than upper limb movement. A 90% accuracy rate may be acceptable for artificial arm control, but is inadequate for safe use of artificial legs because any error may cause the user to fall.

(2) To realize the NMI that can be carried by leg amputees, compact integration of software and hardware on an embedded system is required. Such embedded system should provide high speed and real time computation of neural decoding algorithm because any delayed decision-making from the NMI may result in unsafe use of prostheses. Streaming and storing multiple sensor data and at the same time deciphering user intent with fast time response to guarantee smooth control of artificial legs is a great challenge to the design of an embedded NMI.

Our previous study has shown the potential of using a computer system to accurately decode EMG signals for neural controlled artificial legs [5]. A special purpose computer system was designed for neural machine interface (NMI). The NMI takes EMG inputs from multiple EMG electrodes mounted on user's lower limbs, decodes user's movement intent by EMG pattern recognition (PR) algorithm, and makes decisions to control prosthetic legs. The PR algorithm generally includes two phases: training and testing. During the training phase, the classifier is trained to learn the patterns of EMG signals when the user performs different tasks. This process

involves intensive signal processing and numerical computations. The trained classifier is then used to predict the user's movement during the real time testing phase. Our previous experiments implemented the real time control algorithm on a commodity embedded microcontroller unit (MCU) and the training algorithm on a PC. EMG signals were segmented and processed by sliding analysis windows. The parameters of the trained classifier were manually loaded into the memory of the MCU before the real time testing phase started. Looking at the existing research, several observations are in order:

(1) The accuracy of the neural deciphering algorithm increases as the window length increases, which in turn increases the computation complexity.

(2) The commodity embedded system can barely meet the requirement of real time testing for a very small window length. Increasing the window length for higher accuracy places overburden on the MCU.

(3) It is even more difficult to incorporate both training and testing algorithms in a compact and efficient way on the existing embedded system.

This paper presents the design and implementation of an FPGA-based special purpose embedded system realizing a neural machine interface for artificial legs. Our new system integrates both the training algorithm and the real time control algorithm on an FPGA. New parallel algorithms tailored to FPGA device are developed to fulfill the requirements of high speed processing of training process and real time processing of control functions.

Our newly designed embedded architecture consists of a serial peripheral interface (SPI) module, a training module, a testing module and a flash memory. The

SPI module interfaces with off board analog-to-digital converters (ADC) to sample EMG signals. The training module implements parallel algorithm for training phase. The parameters of the trained classifier are stored in the flash memory. The testing module provides decoding algorithm in real time and sends out decisions for prosthetic leg control.

Our experimental results on the Altera Stratix II GX EP2SGX90 PCI Express platform have shown that the training performance is 3X faster than the software implementation running on a CPU at 2.0 GHz while the power consumption is only tenth of the software version. The real time controlling process achieves a speedup of around 280X over our previous software implementation based on Freescale's MPC5566 MCU with no sacrifice of computation accuracy.

This paper makes the following contributions:

- Design and implementation of an FPGA-based special purpose embedded system for neural machine interface for artificial legs;
- Design of a parallel EMG pattern recognition algorithm specifically tailored to FPGA device to decode user intent in real time;
- Demonstration of the feasibility of a self-contained, high performance, and low power real-time NMI for artificial legs.

2.2 Embedded System Architecture

The overall structure of our embedded system architecture for neural machine interface is shown in Figure 2.1. Multiple channels of EMG signals are collected from different muscles on patient's residual limb using surface electrodes. EMG amplifiers then filter the signal noises and make the polarity and the amplitude range of the EMG

signals compatible with the input requirements of analog-to-digital converters (ADC). The outputs of the amplifiers are converted to digital signals by the ADCs and then streamed to the FPGA device. The sampled digital data are stored in the SRAM of the FPGA chip and segmented by sliding analysis windows. EMG features that characterize individual EMG signals are extracted in each analysis window. The FPGA system works in two modes: training mode and real time testing mode. In the training mode, a large amount of EMG data is collected and processed to train the EMG pattern classifier. The parameters of the trained classifier for later use in the testing mode are stored in the flash memory upon completion of the training phase. In the testing mode, the EMG input streams are processed continuously and sent to the trained classifier for a decision that identifies the user's locomotion mode for every analysis window.

2.3 Pattern Recognition Algorithm

In order to decipher the intended lower limb movements using leg EMG signals, we need first extract EMG features from raw EMG signals and then employ an EMG

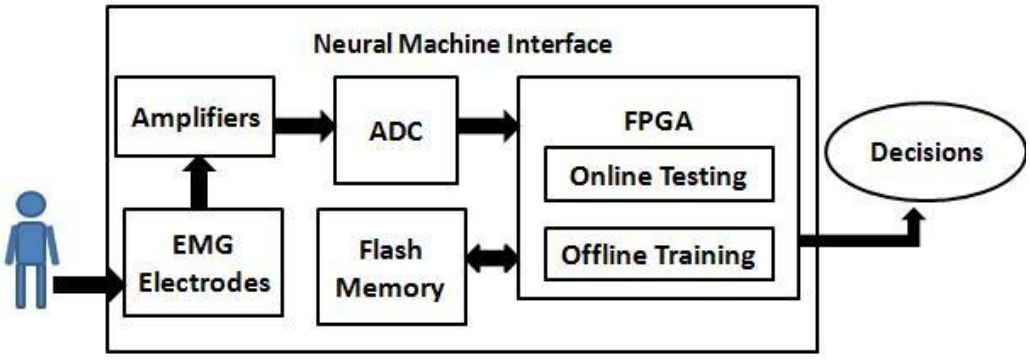

Figure 2.1. Structure of EMG-based neural machine interface for artificial legs.

pattern classifier to identify the movement modes. In this study, EMG time-domain (TD) features and a linear discriminant analysis (LDA) classifier [6] are adopted for our design.

2.3.1 EMG Feature Extraction

Several types of EMG features have been studied for EMG PR, such as time-domain features [6], frequency-domain features [7], and time-frequency features [8]. In our study, four time-domain (TD) features (the mean absolute value, the number of zero crossings, the waveform length, and the number of slope sign changes) are used. In both training procedure and testing procedure, EMG signals are segmented into a series of analysis windows with a fixed window length and a window increment. In an analysis window, four TD features are extracted from the EMG signals for each channel. A $1\times 4N$ feature vector $\bar{f}$ for one analysis window is formulated as $\bar{f}=\{f_1, f_2,...f_n,...f_N\}^T$, where f_n denotes four TD features of channel n, and N denotes the number of EMG channels. In the training procedure, EMG data are recorded for a period of time under each interested movement mode. A $4N\times M$ feature matrix is formed as input to train the EMG pattern classifier, where M is the total number of training windows. In the testing procedure, the feature vector $\bar{f}$ derived from every analysis window is input to the classifier and a movement mode (class) is predicted for each window.

2.3.2 EMG Pattern Classification

Various classification methods have been applied to EMG PR, such as LDA [6], multilayer perceptron [9], Fuzzy logic [10], and artificial neural network [4]. In our

study, a LDA classifier is adopted to decode the movement class. The movement classes are expressed as $C_g (g \in [1, G])$, where G denotes the total number of classes.

Using the linear discriminant analysis, the observed data is classified to the movement class C_g where the posteriori probability $P(C_g \mid \bar{f})$ can be maximized [11]. Given movement class C_g, the observed feature vectors have a multivariate normal (MVN) distribution. Suppose μ_g is the mean vector and $\sum_g$ is the covariance matrix of class C_g, we have $P(\bar{f} \mid C_g) \sim MVN(\mu_g, \sum_g)$. By assuming that every class shared a common covariance, i.e. $\sum_g = \sum$, the movement class can be estimated as $\tilde{C}_g = \arg\max C_g \{d_{C_g}\}$, where $d_{C_g} = \bar{f}^T \sum^{-1} \mu_g - \frac{1}{2} \mu_g{}^T \sum^{-1} \mu_g$ is the linear discriminant function.

During the training procedure, $\sum$ and μ_g are estimated by the feature vectors calculated from the training data as $\tilde{\sum} = \frac{1}{G} \sum_{g=1}^{G} \frac{1}{K_g - 1} (F_g - M_g)(F_g - M_g)^T$ and $\tilde{\mu}_g = \frac{1}{K_g} \sum_{k=1}^{K_g} \bar{f}_{C_g,k}$. K_g is the number of observations in class C_g; $\bar{f}_{C_g,k}$ is the k^{th} observed feature vector in class C_g; $F_g = [\bar{f}_{C_g,1}, \bar{f}_{C_g,2}, ... \bar{f}_{C_g,k}, ... \bar{f}_{C_g,K_g}]$ is the feature matrix; $M_g = [\tilde{\mu}_g, \tilde{\mu}_g, ... \tilde{\mu}_g]$ is the mean matrix that has the same number of columns as in F_g. Therefore, d_{C_g} can be estimated as

$$\tilde{d}_{C_g} = \bar{f}^T \tilde{\sum}^{-1} \tilde{\mu}_g - \frac{1}{2} \tilde{\mu}_g{}^T \tilde{\sum}^{-1} \tilde{\mu}_g . \quad (2.1)$$

Two weight matrices $W = [W_1, W_2, ... W_g, ... W_G]$ and $D = [D_1, D_2, ... D_g, ... D_G]$ are stored

in the flash memory once the training procedure completes. Here $W_g = \tilde{\Sigma}^{-1} \tilde{\mu}_g$ and $D_g = -\frac{1}{2} \tilde{\mu}_g{}^T \tilde{\Sigma}^{-1} \tilde{\mu}_g$. Equation (2.1) can be expressed as

$$\tilde{d}_{C_g} = \bar{f}^T W_g + D_g . \tag{2.2}$$

In the testing phase, the observed feature vector $\bar{f}$ derived from each analysis window is applied to calculate $\tilde{d}_{C_g}$ in (2.2) for each movement class and is classified into a class $\tilde{C}_g$ that satisfies

$$\tilde{C}_g = \arg\max C_g \{\tilde{d}_{C_g}\}, C_g \in \{C_1, C_2, ..., C_G\} . \tag{2.3}$$

2.4 System Implementation

Our system implementation is based on Altera Stratix II GX EP2SGX90 FPGA. The FPGA device is integrated on a PCI Express development board that has 64-Mbyte flash memory. The parallelism of FPGAs allows for high computational throughput even at a low clock rates. The flexibility of FPGAs allows for even higher performance by trading off precision and range in the data format. In this study, we have designed a special PR algorithm to make the best use of FPGA. The PR algorithm is implemented with the help of Impulse C C-to-HDL CoDeveloper software. To optimize the system performance, large tasks have been divided into small function processes. The processes can work in parallel unless there are data dependencies. The communications between different processes can be easily achieved using streams and signals. A stream is a dual-port first-in-first-out (FIFO) RAM buffer, where a producer process stores data and a consumer process loads data. Each process can read data from and write data to multiple streams. Signals are used

to communicate status information from one process to another.

The hardware design and program implementation details are discussed in the following subsections.

2.4.1 Hardware Design

The hardware design of the embedded system is shown in Figure 2.2. The FPGA system is built using Altera SOPC Builder (System on a Programmable Chip Builder) software. All the hardware components on the FPGA are connected by Altera's Avalon memory mapped (Avalon-MM) interface, which is an address-based read/write interface of master-slave connections. In the system, each component has its own memory space. The components with master ports ("M" in Figure 2.2) can

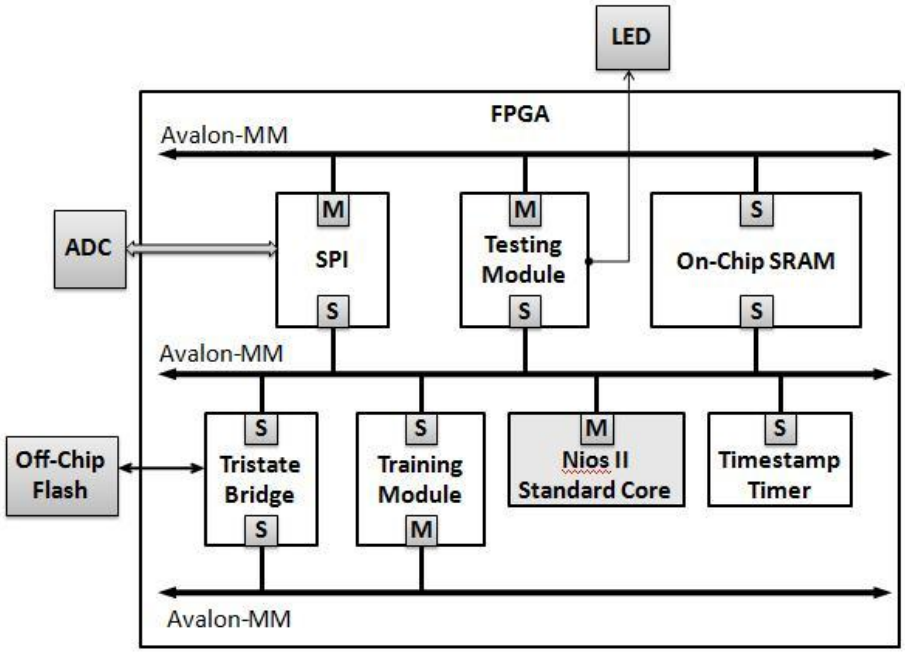

Figure 2.2. Block diagram of embedded system design based on Altera Stratix II GX FPGA device. Avalon-MM: Avalon memory mapped interface; M: master port; S: slave port; ADC: analog-to-digital converter; SPI: serial peripheral interface; SRAM: internal static RAM.

directly read/write data from/to the components with slave ports ("S" in Figure 2.2) if the ports are connected by the same Avalon-MM interface.

The serial peripheral interface (SPI) module communicates with the external ADC chips and controls the EMG data sampling and storing. Due to the parallelism of FPGAs, multiple EMG channels can be sampled and processed in parallel, which effectively speeds up the system performance and reduces complexity of the decoding algorithm. The sampled EMG data are stored in the on-chip SRAM. A special circular buffer is designed for optimizing memory utilization. The training module and the testing module are both user defined function modules that implement the PR algorithms for the training phase and the testing phase, respectively. The training data can be online collected or pre-loaded into the off-chip flash memory before the training process starts. In our experimental results we evaluate the performance of the training phase assuming the training data is already in the flash memory of the board. When the training phase completes, the weight matrices of the EMG pattern classifier are stored in the flash memory for future use in testing phase. During the real time testing procedure, the decision of the EMG classifier is displayed by eight user LEDs. A Nios II/s standard soft processor is used to direct data flows and evaluate the performance of PR algorithms. A timestamp timer is built for the purpose of measuring program execution time. In our design, direct memory access (DMA) is efficiently used for transferring data blocks between user defined modules and memories without the involvement of the Nios II processor. The testing module and the SPI module can directly access memory blocks in the SRAM. The training module can access the flash memory directly.

2.4.2 Task Parallelism and Pipelining

The experimental results of our previous software implementation have shown that the processing of extracting features from multiple channels of EMG signals was the most computation-intensive task for both training and testing procedures. The feature extraction task accounted for more than 90% of the total execution time. Fortunately, we have found that in the feature extraction process, each channel has the identical and independent procedure: sample and store EMG data, load data of one analysis window from memory, calculate the mean of data, subtract the mean from raw data, calculate four TD features and send these features to the LDA classifier. The parallelism of FPGAs enables all the EMG channels to be processed in parallel, rather than in sequence as in the software implementation. Therefore, if there are N EMG channels, N feature extraction threads will be generated.

1) Testing Phase: Figure 2.3 demonstrates high level task stages and data flows of EMG PR for testing phase. Each white box in the figure represents a function process. N parallel feature extraction threads are generated. Each thread contains three stages (processes): *a)* fetch data from memory ("Read_ch$n$"), *b)* calculate the mean of data and subtract the mean from raw data ("SubtractMean_ch$n$"), and *c)* extract four TD features from pre-processed data ("FeatureExtract_ch n "). Within the "FeatureExtract_ch$n$" process, the operations of extracting each feature (the mean absolute value, the number of zero crossings, the waveform length, and the number of slope sign changes) can also be parallelized since they do not depend on each other. A software process running on the Nios II processor triggers the first stage of each thread by sending out a signal ("Start_Chn_Sig") to indicate that EMG data for channel n is

currently available to fetch from memory ("Data_Mem"). The subsequent two stages

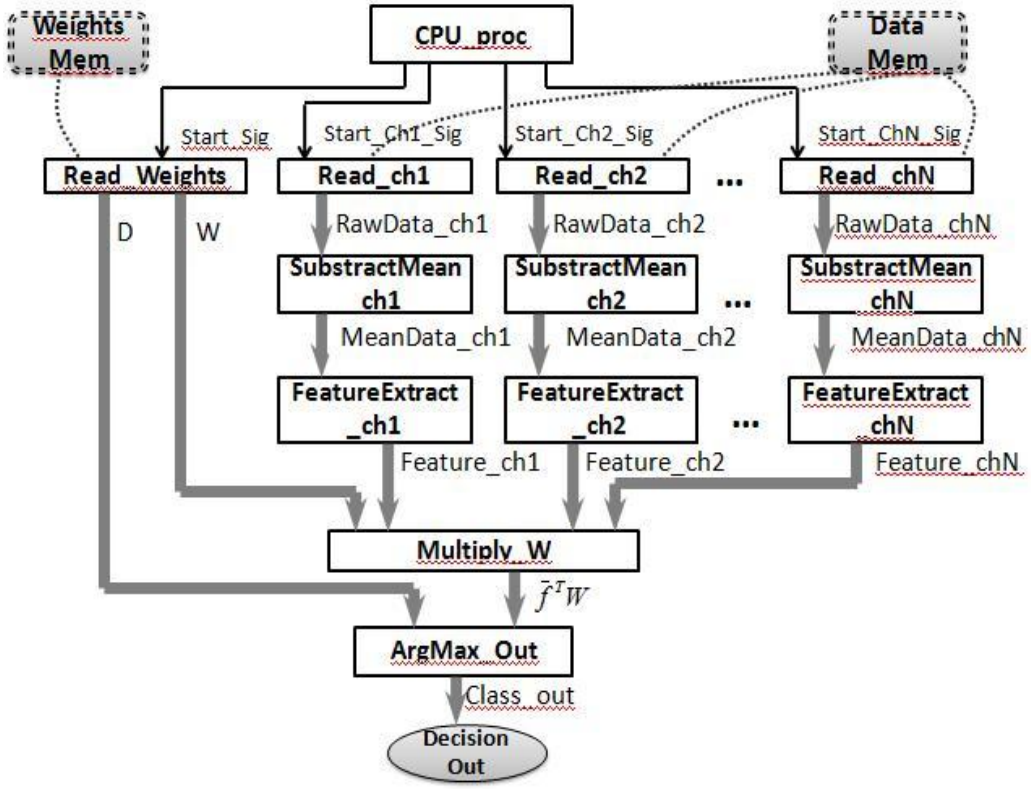

Figure 2.3. Task stages and data flows of EMG PR in the testing phase.

are triggered once there are valid data coming in.

The implementation of LDA classifier also contains three processes (algorithm details in Section 2.3.2): *a)* load weight matrices *W* and *D* from off-chip flash memory ("Read_Weights"), *b)* concatenate four TD features of each channel into a feature vector $\bar{f}$ and calculate $\bar{f}^T W$ ("Multiply_*W*"), *c)* add weight matrix *D* to $\bar{f}^T W$ and output the movement class according to equation (2.3) in Section 2.3.2 ("Argmax_Classify").

2) Training Phase: In the training process, the input to the LDA classifier is an EMG feature matrix composed of a large number of feature vectors extracted from a

series of sliding windows. In addition to the task parallelism among different channels, the feature extractions of subsequent windows can be efficiently pipelined. When the former analysis window has completed the stage of data fetching and moved to next stage to process the data, the following window can begin to fetch data from the memory.

The implementation of LDA classifier for training phase has five stages as shown in Figure 2.4: *a)* sort feature vectors into groups according to movement class, *b)* within each group, calculate mean feature vector and subtract the mean vector from all the feature vectors, *c)* compute covariance matrix of each class, *d)* compute inverse of mean covariance matrix, *e)* compute weight matrices W and D. Wherein, stage *b* and *c* can be parallelized among different groups.

2.4.3 Memory Utilization and Timing Control

In a real time embedded system design, precise timing control and efficient

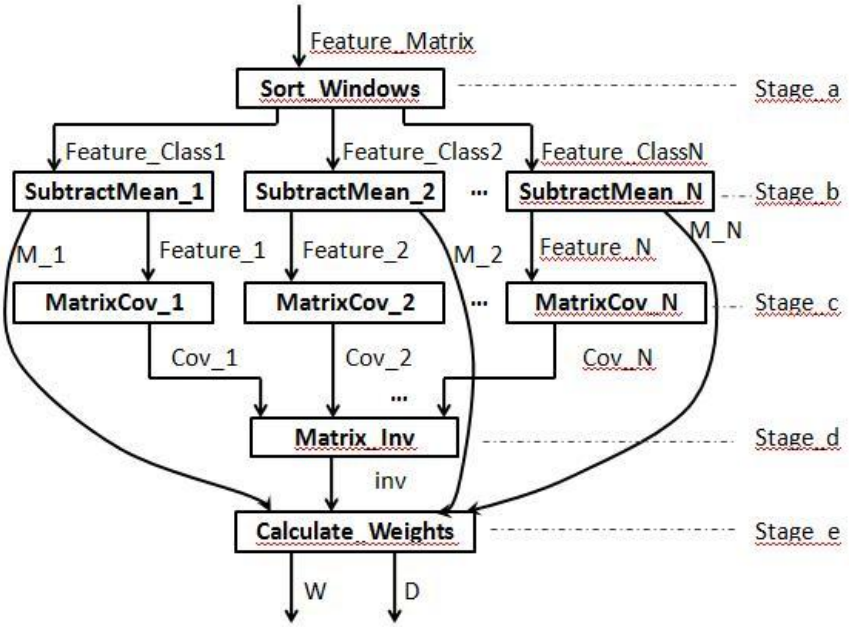

Figure 2.4. Task stages and data flows of LDA classification in the training phase.

memory utilization are very important. To ensure a smooth control of artificial legs, a circular buffer is designed to allow simultaneous data sampling and decision making. Because all the EMG channels are sampled and processed in parallel, each channel has its own buffer. The circular buffer of one channel consists of $Q+1$ memory blocks, where Q is the quotient of window length divided by window increment. Here we choose a proper window length and a window increment to make sure Q is an integer. Each memory block stores the data sampled in one window increment.

Figure 2.5 shows an example to demonstrate the timing control and memory management of our real-time decoding algorithm for one channel. In the example, the window length and the window increment are set to 140ms and 20ms, respectively. Therefore, 140/20+1=8 memory blocks are generated in this buffer. Usually the data sampling frequency is 1000Hz and each sample is a 16-bit data, so the circular buffer size is 8*20*16 bits = 320 bytes.

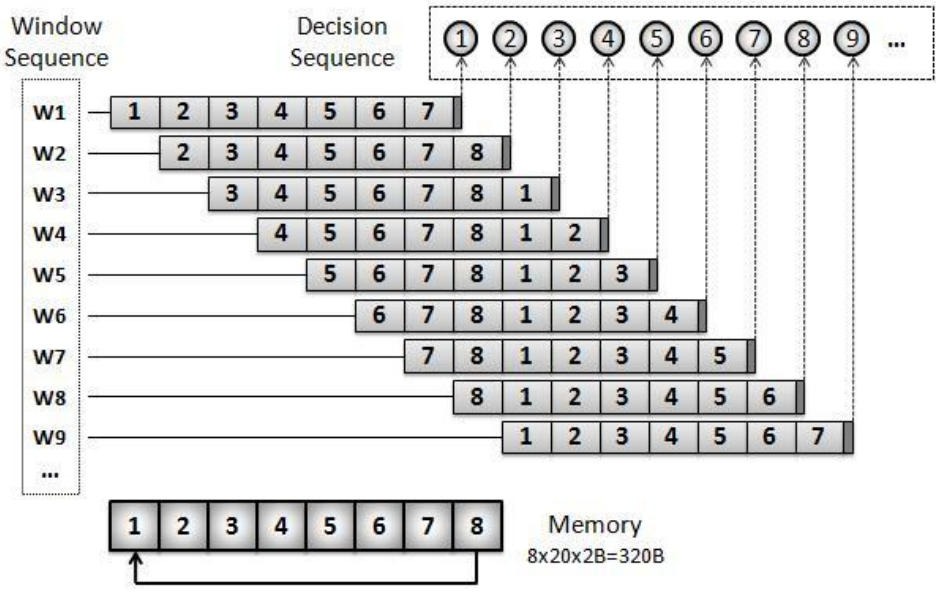

Figure 2.5. Timing control and memory management of real-time control algorithm for one channel

The buffer is divided into 8 blocks. To fill up each block, it takes 20ms. To make the first decision, the system needs to wait for 140ms to get data for the first window $W1$. When blocks 1 to 7 are filled up, the data for $W1$ are available. At this point, PR module can start to fetch data to its local buffer, and the system keeps sampling new data and stores the incoming data in block 8. When block 8 is filled up, the data for the second window $W2$ (blocks 2 to 8) are available and the PR module starts to process $W2$. At this point, the PR computation of $W1$ has been completed and decision 1 has been sent out, so the data in block 1 is no longer useful. Block 1 can be replaced by new data. When processing window $W3$, the data in blocks 3, 4, 5, 6, 7, 8 and 1 are fetched. In this way, the PR algorithm computation and the data sampling are operating in parallel. Since the execution time of PR algorithm is much shorter than 20ms, the system is sufficient to generate a decision every 20ms.

2.4.4 Mixed-Precision Operation

For high accuracy, most computations in LDA algorithm are based on floating point data format. Compared to floating point operations, fixed-point operations result in much higher performance for computation-intensive applications on FPGAs because of their reduced resource cost and lower latency [12]. However, lower-precision operations may lose computation accuracy. To address this problem, mixed-precision operation is employed in our implementation. Since EMG signals are sampled by 16-bit ADCs, they can be represented by 32-bit fixed-point data types without loss of information. We analyzed the arithmetic operation types in the PR algorithm and found that most operations only involved EMG data summation, subtraction, and multiplication/division by integers, all can be handled by fixed-point

data formats with careful tracking except for the computation of covariance matrix and inverse matrix. Therefore in our design, floating point data type is only used during the computations of covariance matrix and inverse matrix in the training procedure.

2.5 Prototyping & Experimental Results

2.5.1 NMI Prototype

Based on the design described in the previous sections, we implemented a prototype board as shown in Figure 2.6. The performance of the FPGA implementation is compared with our previous software implementation using a combination of a PC and a commodity embedded system [5]. On our FPGA implementation, both the training algorithm and the real time control algorithm are implemented on Altera Stratix II GX EP2SGX90 FPGA PCI Express platform. In the software implementation, training phase is run on the PC with 2 GHz AMD Turion 64 X2 CPU while the real time testing phase is run on the embedded system using Freescale's MPC5566 132 MHz 32 bits microcontroller with Power Architecture.

Figure 2.6. The prototype board based on Altera Stratix II GX EP2S90 PCI Express platform.

Both NMI prototypes are used to recognize the user's intended movement transitions between sitting and standing, two basic but difficult tasks for patients with transfemoral amputations. Seven EMG channels are applied in the prototypes. Two movement classes are considered: sitting and standing. Four TD features and a LDA-based classifier discussed in previous sections are used in the PR algorithm. Overlapped analysis windows are used in order to gain timely system response. The window length and the window increment are set to 140ms and 20ms, respectively, with the EMG data sampling frequency of 1000Hz. The system clock of the FPGA NMI prototype is 100MHz. The resource utilization summary is presented in Table 2.1.

2.5.2 Experimental Settings and Results

Our NMI prototype contains a serial peripheral interface (SPI) module that is designed to interface with off board ADCs for data sampling. Since our current development board does not have enough IOs to interface with multiple EMG channels, we set up a two-phase experiment to evaluate the performance of our real-time control algorithm.

1) Evaluations of computation accuracy and speed: To evaluate the computation

Table 2.1. Stratix II GX EP2SGX90 resource utilization

Resource	Training	Testing
ALUTs (72,768)	17,763 (24%)	21,857 (30%)
Registers (72,768)	18,414 (25%)	19,929 (27%)
Mem bits(4,520,448)	1,876,343(41%)	1,210,530 (27%)
DSP blocks (384)	222 (58%)	184 (48%)

accuracy of our fixed-point computation on the FPGA implementation, our experiment was based on the same testing dataset as the software floating-point implementation. The testing dataset was preloaded into the memory. The experiment results show that given a series of 400 analysis windows, the decision sequence of the FPGA system was exactly the same as the software implementation.

To generate one decision, the average execution time of our parallel implementation running on EP2SGX90 was 0.283ms. Compared this value with the performance of our real-time software implementation (80ms) based on Freescale's MPC5566 132 MHz 32 bits microcontroller with Power Architecture, the new design using FPGA implementation gives a 280X speedup.

2) Evaluation of real time control: Although there were no real EMG data sampled in, we assumed the circular buffer kept receiving new data. The PR module was triggered by a countdown counter every 20ms. Once triggered, the PR module began to fetch data from the circular buffer and process the data. A debugging software program was running on the Nios II processor. The result shows that a decision-out message was displayed on the Nios II software console every 20ms, which means that the real time decision algorithm sent out a decision every 20ms continuously. The result indicates a smooth and prompt real-time control. Once we move the design to a new board with enough IOs, the PR algorithm will be triggered by a signal as soon as one memory block of the circular buffer has been filled up (details in Section III). The decision making time interval is still 20ms.

Table 2.2 compares the execution time of the training procedure for different window sizes between the software and the FPGA implementations. The software

Table 2.2. Training execution times and speedups

Window size	Software on PC	FPGA	Speedup
100	87.28 ms	28.93 ms	3.02
200	177.29 ms	56.94 ms	3.11
400	356.64 ms	112.94 ms	3.15

version was developed using Microsoft VC++ and ran on AMD Turion 64 X2 CPU at 2.00 GHz. From the table we can see the FPGA implementation gives a speedup around 3X over the software implementation.

Low power consumption is very important to the final implementation of the NMI for artificial legs. The power consumption of our FPGA implementation measured by Altera PowerPlay Early Power Estimator was 3.499 W. The AMD Turion 64x2 processor has a reported thermal design power of 35 W, which is about 10 times higher than our implementation.

2.6 Conclusions

This paper presented a new design and implementation of an FPGA-based neural-machine interface (NMI) for artificial legs. The NMI implemented both the computation-intensive training algorithm and the real time controlling algorithm on board. The implementation of the training algorithm achieved a speedup of 3X over the software implementation running on a 2.0 GHz CPU, and only consumed tenth of power. The performance of FPGA-based real time decision algorithm was 280X faster than the software implementation on Freescale's MPC5566 microcontroller with no sacrifice of computation accuracy. The high efficiency and accuracy were achieved through parallelized and pipelined algorithm, mixed-precision operations, precise timing control, and optimized memory utilization. The results demonstrated the

feasibility of a self-contained, low power, and high performance real-time NMI for artificial legs. Our future work includes implementation of an embedded system that fuses the information of EMG signals and kinematics for deciphering other lower limb movement tasks such as walking on different terrains and testing our NMI on leg amputees.

List of References

[1] J. V. Basmajian and C. J. De Luca, *Muscles alive: their functions revealed by electromyography*, 5th ed. Baltimore: Williams & Wilkins, 1985.

[2] P. A. Parker and R. N. Scott, "Myoelectric control of prostheses," *Crit Rev Biomed Eng*, vol. 13, pp. 283-310, 1986.

[3] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans Biomed Eng*, vol. 50, pp. 848-54,2003

[4] H. Huang, T. A. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Trans Biomed Eng*, vol. 56, pp. 65-73, 2009.

[5] H. Huang, Y. Sun, Q. Yang, F. Zhang, X. Zhang, Y. Liu, J. Ren and F. Sierra, "Integrating Neuromuscular and Cyber Systems for Neural Control of Artificial Legs," *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems,* 2010, pp. 129-138.

[6] B. Hudgins, P. Parker, and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Trans Biomed Eng*, vol. 40, pp. 82-94, 1993.

[7] M. Zecca, S. Micera, M. C. Carrozza, and P. Dario, "Control of multifunctional prosthetic hands by processing the electromyographic signal," *Crit Rev Biomed Eng*, vol. 30, pp. 459-85, 2002.

[8] W. Xiao, H. Huang, Y. Sun, and Q. Yang, "Promise of embedded system with GPU in artificial leg control: Enabling time-frequency feature extraction from electromyography," *Conf Proc IEEE Eng Med Biol Soc*, vol. 1, pp. 6926-9, 2009.

[9] N. F. Guler and S. Kocer, "Classification of EMG signals using PCA and FFT," *J Med Syst*, vol. 29, pp. 241-50, 2005.

[10] A. B. Ajiboye and R. F. Weir, "A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control," *IEEE Trans Neural Syst Rehabil Eng*, vol. 13, pp. 280-91, 2005.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. New York: Wiley, 2001.

[12] J. Sun, G. D. Peterson, O. O. Storaasli, "High-Performance Mixed-Precision Linear Solver for FPGAs", *IEEE Trans Computers*, vol. 57, pp. 1614-1623, 2008.

MANUSCRIPT 3

Implementing an FPGA System for Real-Time Intent Recognition for Prosthetic Legs

by

[1]Xiaorong Zhang, He Huang, and Qing Yang

is published in *the processing of the 49th Design Automation Conference (DAC'12)*,

San Fransico, CA, 2012. p. 169-175.

[1] Xiaorong Zhang, He Huang, and Qing Yang are with Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, 02881, Email {zxiaorong, huang, qyang}@ele.uri.edu.

Abstract

This paper presents the design and implementation of a cyber physical system (CPS) for neural-machine interface (NMI) that continuously senses signals from a human neuromuscular control system and recognizes the user's intended locomotion modes in real-time. The CPS contains two major parts: a microcontroller unit (MCU) for sensing and buffering input signals and an FPGA device as the computing engine for fast decoding and recognition of neural signals. The real-time experiments on a human subject demonstrated its real-time, self-contained, and high accuracy in identifying three major lower limb movement tasks (level-ground walking, stair ascent, and standing), paving the way for truly neural-controlled prosthetic legs.

3.1 Introduction

Neural-machine interface (NMI) is a typical example of biomedical cyber physical system (CPS) which utilizes neural activities to control machines. The neural signals collected from nerves, central neurons, and muscles contain a lot of important information that can represent human states such as emotion, intention, and motion. In such a CPS, a computer senses bioelectric signals from a physical system (i.e. human neural control system), interprets these signals, and then controls an external device, such as a power-assisted wheelchair [1], a telepresence robot [2], or a prosthesis [3-5], which is also a physical system.

The neural signals captured from muscles are called electromyographic (EMG) signals. The EMG signals can be picked up with electrodes on the body surface and are effective bioelectric signals for expressing movement intent. In recent years, EMG-

based NMI has been widely studied for control of artificial limbs in order to improve the quality of life of people with limb loss.

Researchers have aimed at utilizing neural information to develop multifunctional, computerized prosthetic limbs that perform like natural-controlled limbs. The NMI needs to interface with multiple sensors for collecting neural signals, decipher user intent, and drive the prosthetic joints simultaneously. EMG pattern recognition (PR) is a sophisticated technique for characterizing EMG signals and classifying user's intended movements. It usually contains a training phase for constructing the parameters of a classifier from a large amount of EMG signals, and a testing phase for recognizing user intent using the trained classifier. While the PR algorithm for artificial arm control has been successfully developed and neural-controlled prosthetic arms have already been clinically tested [4, 6-7], there has been no EMG-based NMI commercially available for control of powered prosthetic legs. Challenges in the management of both physical and computational resources have limited the success of a CPS for neural control of artificial legs.

One of the challenges on physical resources is due to the muscle loss of leg amputees. Patients with leg amputations may not have enough EMG recording sites available for neuromuscular information extraction [8]. The non-stationary of EMG signals during dynamic leg movement further increases the difficulty of user intent recognition (UIR). To address this challenge, Huang et al. proposed a phase-dependent PR strategy for classifying user's locomotion modes [8]. This PR algorithm extracted neural information from limited signal sources and showed accurate classification (90% or higher accuracy) of seven locomotion modes when 7-9 channels of EMG signals

were collected from able-bodied subjects and leg amputees. The performance of the phase-dependent PR strategy was further improved by incorporating EMG signals with mechanical signals resulting from forces/moments acting on prosthetic legs [9]. The experimental results showed that the classification accuracies of the neuromuscular-mechanical fusion based PR algorithm were 2%-27% higher than the accuracies derived from the strategies using EMG signals alone, or mechanical signals alone [9].

The challenges on computational resources include tight integration of software and hardware on an embedded computer system that is specifically tailored to this environment. It requires high speed classifier training, fast response, real-time decision making, high reliability, and low power consumption. Embedded systems are usually resource constrained and typically have processors with slower system clock, limited memory, small or no hard drives. To make the idea of neural-controlled artificial leg a reality, we need efficiently manage the constrained computational resources to meet all the requirements for smooth control and safe use of prosthetic legs. Our previous study proposed an NMI implemented on a commodity 32-bit microcontroller unit (MCU) for recognizing two non-locomotion tasks of sitting and standing in real-time [10]. It was reported that there was a noticeable delay of 400 ms for producing classifying decisions, implying inadequate computational power of the MCU for real-time control of artificial legs. Furthermore, this NMI implementation realized only the testing phase of the PR algorithm on the MCU. The training algorithm which involved intensive computations was implemented on graphic processing units (GPUs) and showed good speedups over CPU-based implementation [10]. However, currently most of the GPU cards only have PCI Express interfaces and are not portable. Relative high power

consumption further makes it more difficult to use GPU as an embedded wearable device.

To tackle these technical challenges, we present a new design of an embedded system that is specifically tailored to the new NMI. A unique integration of hardware and software of the embedded system is proposed that is suitable to this real-time CPS with adequate computational capability, high energy efficiency, flexibility, reliability, and robustness. The NMI on an embedded platform continuously monitors EMG activities from leg muscles as well as mechanical forces/moments acting on prosthetic legs. Information fusion technique is then used to decode and decipher the collected signals to recognize users' intended locomotion modes in real-time. The embedded system contains two major parts: a data collection module for sensing and buffering input signals and an intelligent processing unit for executing the UIR algorithm. The data collection module was implemented on a microcontroller unit (MCU) with multiple on-board analog-to-digital converters (ADCs) for signal sampling. A reconfigurable FPGA device was designed as the main computing engine for this system. There are several reasons for choosing FPGAs for the designed NMI. First, the parallelism of FPGAs allows for high computational throughput even at low clock rates. Secondly, FPGAs are not constrained by a specific instruction set, thus are more flexible and more power efficient than processors. Furthermore, FPGAs can easily generate customized IO interfaces with existing IP cores, and appear to be good choices for real-time embedded solutions. In our design, a high-level synthesis tool was used to help reducing the implementation difficulty of coding with hardware design language (HDL). A special parallel processing algorithm for UIR was designed, realizing the

neuromuscular-mechanical fusion based PR algorithm coupled with the real-time controlling algorithm in hardware. A serial peripheral interface (SPI) was built between the MCU and the FPGA to transfer digitized input data from the MCU to the FPGA device. The decision stream of user's intended movements can be output to either control a powered prosthetic leg or drive a virtual reality (VR) system with the purpose of evaluating the NMI. Although our previous research has made the attempt to use FPGA in EMG pattern recognition and has shown high processing speed in the offline analyses [11], the embedded system presented here is the first complete CPS for the NMI that implements both training and testing modules on one single chip. The New CPS integrates all the necessary interfaces and control algorithms for interacting with the physical system in real-time.

The newly designed NMI was completely built and tested as a working prototype. The prototype was then used to carry out real-time testing experiments on an able-bodied subject for classifying three movement tasks (level-ground walking, stair ascent, and standing) in real-time. The system performance was evaluated to demonstrate the feasibility of a self-contained and high performance real-time NMI for artificial legs. Videos of our experiments on the human subject can be found at http://www.youtube.com/watch?v=KNhihjXProU.

This paper is organized as follows. Next section presents the overall system design. Section 3.3 describes the detailed implementation of the UIR algorithm. The experimental results are demonstrated in Section 3.4. We conclude our paper in Section 3.5.

3.2 System Design

The architecture of designed CPS is shown in Figure 3.1. The embedded NMI samples input signals from two physical systems--a human neuromuscular system and a mechanical prosthetic leg. The sampled signals are then processed to decipher user's intent to control the prosthesis. The NMI consists of two modules: a data collection module built on an MCU with multiple on-chip ADCs for sensing and buffering input signals, and an FPGA device as the computing engine for fast data decoding and pattern recognition. A serial peripheral interface (SPI) is located between the two devices for transferring digitized input data from the MCU to the FPGA device.

1) Input signals: Multi-channel EMG signals are collected from multiple surface electrodes mounted on patient's residual muscles. Mechanical forces and moments are recorded from a 6 degrees-of-freedom (DOF) load cell mounted on the prosthetic pylon. The EMG signals and the mechanical signals are preprocessed by filters and amplifiers and then simultaneously streamed into the NMI.

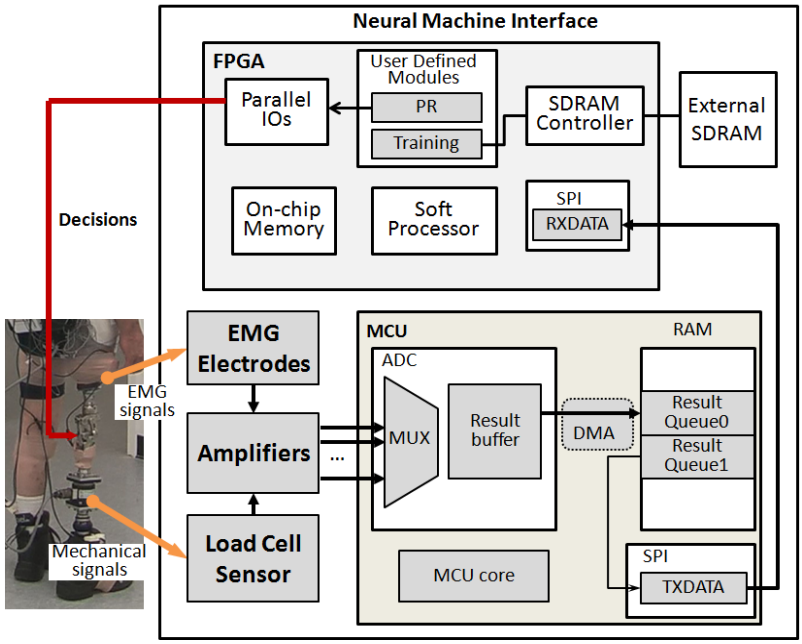

Figure 3.1. System architecture of the embedded NMI for artificial legs.

2) MCU module: The MCU device does not do any compute-intensive task. It provides multi-channel on-chip ADCs to sample the input signals and convert the analog signals to digital data. The digitized data is then stored in the user-defined result queues allocated in the RAM buffer. In the system RAM, two equal sized result queues are defined. With direct memory access (DMA) support, the MCU core can be insulated from the data acquisition process. Thus these two result queues forms a circular buffer that can continuously receive new data while transmitting old data to the FPGA module for further processing.

3) FPGA module: The FPGA device receives digitized data from the MCU module continuously. In order to fully utilize the computing capacity of the FPGA system and produce dense decisions, the input signals are segmented by overlapped analysis windows with a fixed window length and window increment [4]. The designed FPGA module contains six components: an SPI module that serially receives input signals from the MCU module, a user defined module implementing the UIR algorithm, a high-speed on-chip memory for fast online pattern recognition, an SDRAM controller that interfaces with a large-capacity external SDRAM, parallel IOs for outputting UIR decisions, and a soft processor for managing hardware components and directing data flows. The FPGA module works in two modes: offline training and online pattern recognition. Offline training needs to be performed before using the artificial leg and also whenever a complete re-training is required. During the training procedure, users are instructed to do different movement tasks, and a large amount of data is collected by the NMI to train the classifier. The external SDRAM is only used in the offline training phase to store the training data because FPGAs usually have

limited on-chip memory. For online pattern recognition, the input streams are stored in the on-chip memory for fast processing and provided to the classifier for decisions to continuously identify the user's intended movements.

3.3 Implementation of the UIR Algorithm on FPGA

3.3.1 Architecture of the UIR Strategy

The architecture of the UIR strategy based on neuromuscular-mechanical information fusion and phase-dependent pattern recognition (PR) is shown in Figure 3.2. It is a self-contained architecture that integrates the functions of training and phase-dependent pattern recognition in one embedded system. For every analysis window, features of EMG signals and mechanical signals are extracted from each input channel. A feature vector is formed and normalized by fusing the features from all the input channels. The feature vector is then fed to the classifier for pattern recognition. The

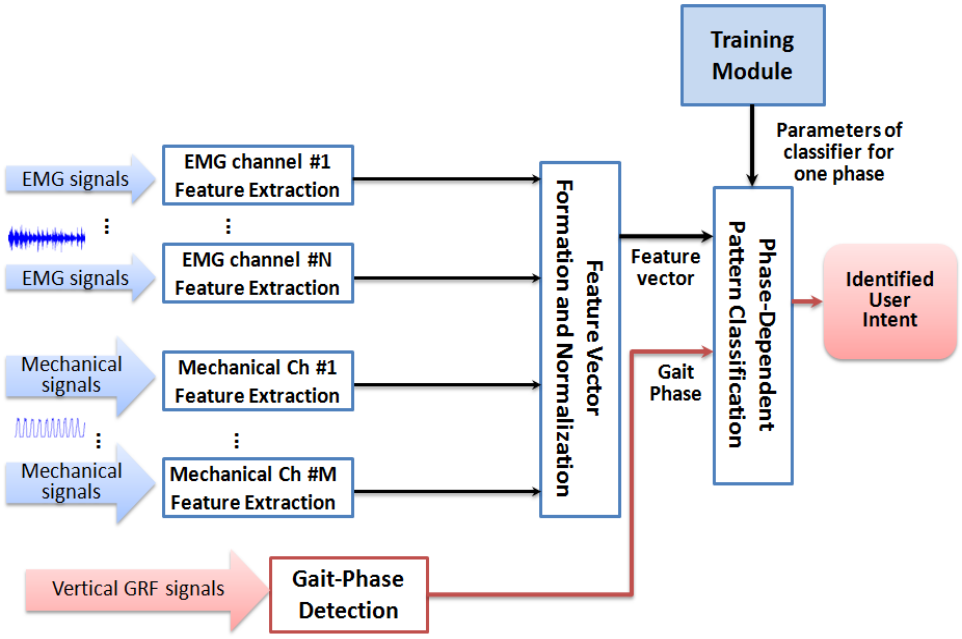

Figure 3.2. Architecture of UIR strategy based on neuromuscular-mechanical fusion and phase-dependent pattern recognition.

phase-dependent classifier consists of a gait phase detector and multiple classifiers. Each classifier is associated with a specific gait phase. During the process of pattern recognition, the gait phase for current analysis window is first determined by the phase detector, and then the corresponding classifier is adopted to do the classification. In this study, four gait phases are defined: initial double limb stance (phase 1), single limb stance (phase 2), terminal double limb stance (phase 3), and swing (phase 4) [9]. The real-time gait phase detection is based on the measurements of the vertical ground reaction force (GRF) sampled from the 6-DOF load cell.

In the real-time embedded system design, to ensure a smooth control of artificial legs, precise timing control is necessary. Figure 3.3 shows the timing diagram of the control algorithm during the real-time UIR process. In the designed system, the MCU and the FPGA device collaborates to produce a decision at every window increment. While the MCU is sampling data for window $i+1$, the user intent recognition for window i, including the tasks of SPI data transfer, feature extraction, gait phase

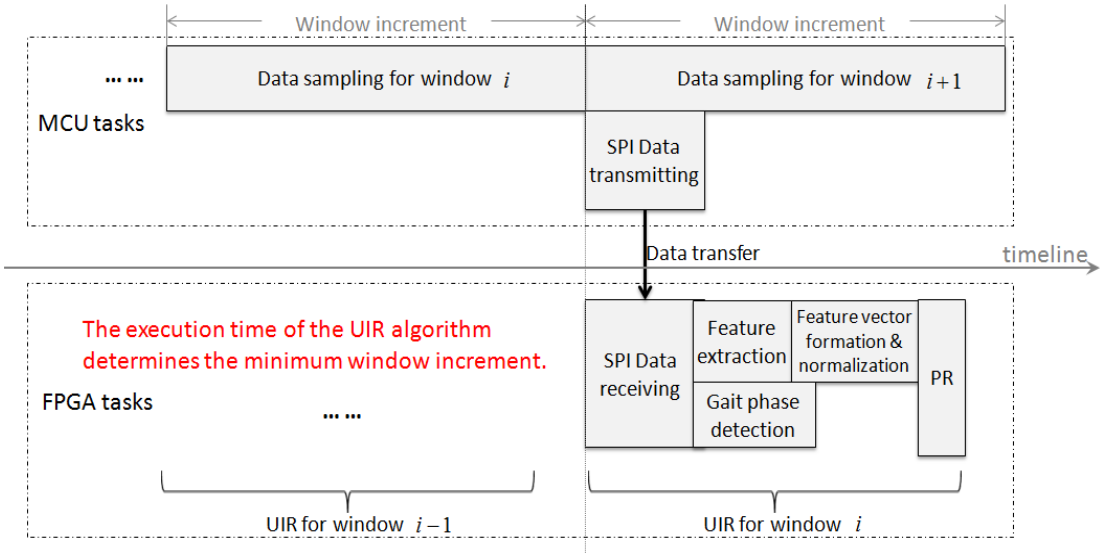

Figure 3.3. Timing diagram of the control algorithm during online UIR process.

detection, feature vector formation and normalization, and pattern recognition must be done within the window increment. In other words, the execution time of the UIR algorithm determines the minimum window increment. Larger window increments will introduce longer delay to the NMI decision, which may not be safe to control the prosthesis in real-time. Therefore fast processing speed is very critical to the embedded system design.

3.3.2 Parallel Implementations on FPGA

The implementation of the CPS was based on the Altera DE3 education board with a Stratix III 3S150 FPGA device, coupled with the Freescale MPC5566 132 MHz 32 bits MCU evaluation board (EVB) with 40-channel 12-bit on-chip ADCs. The MPC5566 module and the DE3 module are connected with each other via serial peripheral interface (SPI). In this design, DE3 was configured as the SPI master and MPC5566 was the slave. A parallel UIR algorithm tailored to FPGA was designed and implemented on DE3. Fixed-point operations were adopted in this implementation because of their less resource cost and lower latency than floating-point operations. In addition, because the input signals were sampled by ADCs with 12-bit resolution, all the arithmetic operation types in the PR algorithm could be handled by 32-bit fixed-point data formats with careful management.

The UIR algorithm was implemented on the FPGA with the help of a high-level synthesis tool--CoDeveloper from Impulse Accelerated Technologies. The PR algorithm was first developed using C programming language, and then CoDeveloper was used to generate VHDL (VHSIC hardware description language) modules from the C program. The VHDL modules were integrated into the FPGA system as the user

defined modules as shown in Figure 3.1, and worked with other hardware components as a complete NMI. To utilize the parallelism of FPGAs, CoDeveloper provides a multiple process, parallel programming model. In our design, the algorithm was partitioned into a set of processes. These processes can run on the FPGA in parallel if there are no data dependencies. The communications between processes can be done using communication objects, such as streams, signals, and shared memories. Streams are implemented in hardware as dual-port FIFO RAM buffers. A stream connects two concurrent processes (a producer and a consumer), where the producer stores data into and the consumer accesses data from the stream buffer. A single process can be associated with multiple input and output streams. Signals are useful objects to communicate status information among processes. Shared memories are used to store and access large blocks of data from specific external memory locations using block read and block write functions.

1) Feature Extraction: Before offline training or online pattern recognition is performed, features need to be extracted from raw input signals. In every analysis window, four time-domain (TD) features (mean absolute value, number of zero crossings, waveform length, and number of slope sign changes) are extracted from each EMG channel. For the mechanical forces/moments recorded from the 6-DOF load cell, the mean value is calculated as the feature from each individual DOF. The procedure of feature extraction is independent for individual input channel and identical for homogeneous sensors. This property can be utilized to greatly reduce the computation time for feature extraction because all the channels can be processed in parallel. Figure 3.4 shows the partitioned processes and the data flows of the FPGA implementation of

feature extraction. Each white box in the figure represents a small process. The black arrows located between processes are one-way data streams. In this design, $N+6$ parallel threads are generated, where N denotes the number of EMG channels, and the other six threads are assigned for extracting features from mechanical forces/moments. For each EMG channel, the thread contains four processes: loading raw input data from memory, calculating mean, subtracting mean from the raw data, and extracting four TD features from the processed data. For mechanical forces/moments, each thread fetches raw data from memory and calculates mean as the mechanical feature. After all the features are extracted, the feature streams are sent to the process of feature vector formation and normalization and then fused into a $(4N+6)\times 1$ feature vector. To implement the phase-dependent PR strategy, a thread of gait phase detection loads the vertical GRF measured from the load cell in each analysis window, and then

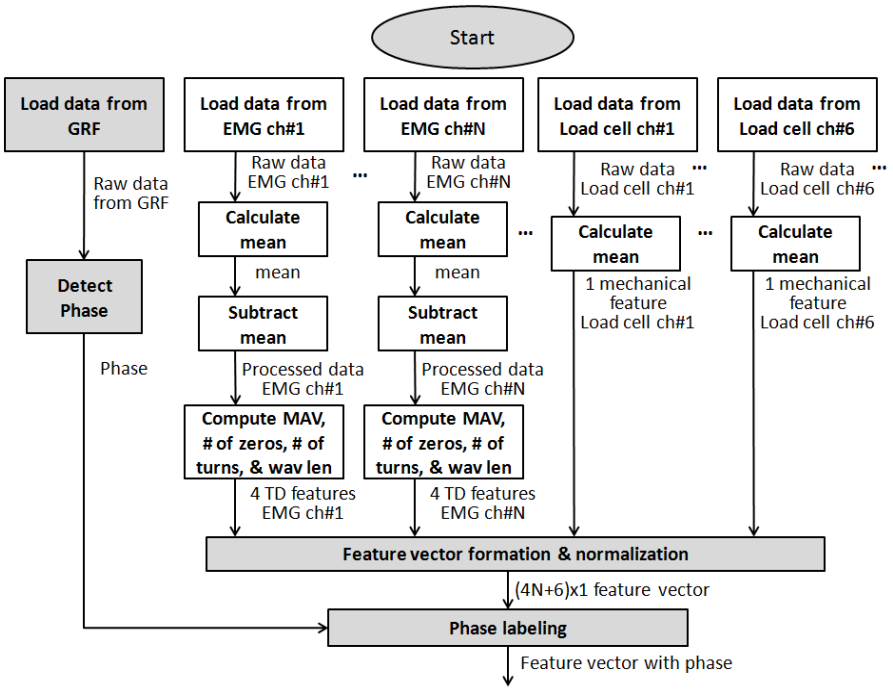

Figure 3.4. Partitioned processes and data flows of the FPGA implementation of feature extraction.

determines current gait phase. This thread is also independent from the threads for feature extraction so that it can run simultaneously with other threads. The detected gait phase is streamed to the phase labeling process, and the feature vector generated in current window is labeled with a specific gait phase. During online pattern recognition, the feature vector with a labeled phase is the input data for pattern classification. In the training procedure, signals are recorded for a period of time under each movement task. Same procedure of feature extraction is performed for every training window. A $(4N+6)\times M_p (p \in [1,4])$ feature matrix is generated as the training data for each gait phase, where M_p is the number of training windows in the p^{th} phase.

2) Pattern Recognition: In this study, linear discriminant analysis (LDA) is adopted for user intent classification because of its computational efficiency for real-time prosthesis control and the comparable accuracy to more complex classifiers [7]. Four gait phases are defined for recognizing user's locomotion mode, giving rise to four LDA-based classifiers. Each classifier is trained for a specific phase. The details of the LDA algorithm can be found in Appendix 3A.

Most of the computations involved in the training algorithm are matrix operations. Because a large amount of data need to be processed in the training procedure, the dimensions of the matrices can be very large. Only using on-chip memory is not enough to handle all the computations. External memory with large capacity is required to store the processing data. In our implementation, several external memory buffers are defined to store either large matrices during the training computations or data that might be reused in the online PR phase or the re-training phase. A process is designed to perform a simple task with a small block of data, such as a matrix row/column, and

store partial results in the external memory. In this way, the operations of subsequent matrix rows/columns can be efficiently pipelined.

During online pattern recognition, based on the gait phase of current analysis window, the parameters of the corresponding classifier are loaded from memory. The observed feature vector derived from each analysis window is provided to the classifier for intent recognition.

3.4 Prototyping & Experimental Results

This study was conducted with Institutional Review Board (IRB) approval at our university and informed consent of subjects. To evaluate the performance of the designed NMI, two experiments with different purposes were conducted. First, to evaluate the classification accuracy and the computation speed of the FPGA-based PR algorithm, the performance of the FPGA implementation was compared with our previous software implementation by processing the same dataset offline. Secondly, to evaluate the performance of the entire CPS, a real-time test was carried out on a male able-bodied subject for identifying three movement tasks (level-ground walking, stair ascent, and standing).

3.4.1 Performance of FPGA vs. CPU

In order to verify the correctness of the FPGA-based PR algorithm and compare the performance of the FPGA design with our previous Matlab implementation, we processed the same dataset on both platforms. The testing dataset was previously collected from a male patient with transfemoral amputation (TF). Seven EMG channels recording signals from the gluteal and thigh muscles and six channels of mechanical forces/moments measured by a 6-DOF load cell were collected in this

dataset for identifying three locomotion modes including level-ground walking, stairs ascent, and stairs descent. The dataset was segmented by overlapped analysis windows. The window length and the window increment were set to 160 data points and 20 data points, respectively. The dataset contained 936 analysis windows totally, where 596 of them were used as the training data and the rest 340 windows were testing data. The Matlab implementation was based on a PC with Intel Core i3 3.2 GHz CPU and 6 GB DDR3 SDRAM at 1333 MHz. For the FPGA implementation, a 1GB DDR2-SDRAM SO-DIMM module was plugged into the DDR2 SO-DIMM socket on the DE3 board as the system external memory. The Altera high performance DDR2 SDRAM IP generated one 200 MHz clock as SDRAM's data clock and one half-rate system clock 100 MHz for all other hardware components in the system. The dataset was preloaded into the SDRAM, and the output decisions were printed to the Nios II console [12] for performance evaluation.

It was observed that the classification results of the FPGA system matched very well with the Matlab implementation. Both platforms provided a training accuracy of 98.99% and a testing accuracy of 98.00%. The missed classification points of the two implementations appeared in the same locations. These results clearly demonstrated that the FPGA-based PR algorithm did not lose any computation accuracy as compared to the software implementation.

Table 3.1 compares the execution time of the LDA-based PR algorithm between the software implementation and the FPGA design. Two configurations with different number of input channels were considered, one with 7 EMG channels and 6 mechanical channels, the other with 12 EMGs and 6 mechanical channels. For the

Table 3.1. Comparison of the execution time of the PR algorithm

	Configuration	FPGA	Matlab	Speedup
Training Algorithm (600 analysis windows)	7 EMGs 6 Mech.	0.46 s	3.2 s	6.96 x
	12 EMGs 6 Mech.	0.64 s	4.7 s	7.34 x
Testing algorithm (classify one analysis window)	7 EMGs 6 Mech.	0.23ms	6.8 ms	29.56 x
	12 EMGs 6 Mech.	0.25 ms	9.5 ms	38.00 x

training algorithm that processed 600 analysis windows, the FPGA provided a speedup of around 7X over the software implementation. In the testing phase, the FPGA system took less than 0.3 ms to classify one analysis window. Compared with the Matlab implementation, the FPGA-based PR testing algorithm demonstrated a speedup of 30 times for the configuration of 7 EMGs and 6 mechanical signals. If more input channels were used (i.e. 12 EMG channels and 6 mechanical channels), a more significant speedup of 38X was observed, which further demonstrated the advantages of FPGA parallelism. From Table 3.1 we can see that the FPGA implementation of the testing algorithm shows better performance than the training algorithm. This is because the testing algorithm only used fast on-chip memory while the computation complexity of the training algorithm required the FPGA to interact with the external memory. In our experiments it was observed that loading training data from external memory to the FPGA took more than half of the total execution time of the training algorithm. The summary of FPGA resource utilization is listed in Table 3.2.

3.4.2 System Performance in Real-Time

Table 3.2. Stratix III 3S150 resource utilization

Resources	**Available**	**Training 12 EMG 6 Mech.**	**Training 7 EMG 6 Mech.**	**Online PR 12 EMG 6 Mech.**	**Online PR 7 EMG 6 Mech.**
Combinational ALUTs	113,600	46%	33%	32%	25%
Memory ALUTs	56,800	3%	2%	3%	2%
Registers	113,600	43%	30%	27%	24%
Block memory bits	5,630,976	16%	12%	16%	12%
DSP blocks	384	72%	44%	27%	24%

The designed NMI prototype was tested on one male able-bodied subject (Figure 3.5) in real-time. A plastic adaptor was made so that the subject could wear a hydraulic passive knee on the left side. Seven surface EMG electrodes (MA-420-002, Motion Lab System Inc., Baton Rouge, LA) were used to record signals from the gluteal and thigh (or residual thigh) muscles on the subject's left leg. An MA-300 system (Motion Lab System Inc., Baton Rouge, LA) collected seven channels of EMG signals. A ground electrode was placed near the anterior iliac spine of the subject. The mechanical ground reaction forces and moments were measured by a 6-DOF load cell mounted on

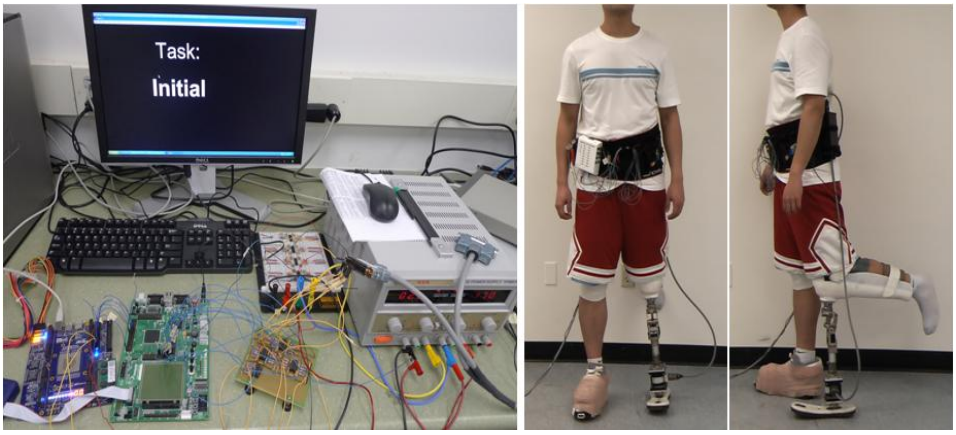

Figure 3.5. The NMI prototype based on MPC5566 EVB and DE3 education board (left figure) and the experimental setup of the real-time test on a male able-bodied subject (right figure).

the prosthetic pylon. The analog EMG signals and mechanical signals were digitally sampled at the rate of 1.1 KHz by the MPC5566 EVB. The intent decisions made by the FPGA device were sent out to 4-bit parallel IO pins on the DE3 board, and displayed by a software GUI. The window length and the window increment were still set to 160 data points and 20 data points, respectively.

Three movement tasks (level-ground walking (W), stair ascent (SA) and standing (ST)) and four mode transitions (ST→W, W→ST, ST→SA and SA→ST) were investigated in this experiment. For the subject's safety, he was allowed to use hand railings and a walking stick. A training session was conducted first to collect the training data for the pattern classification. The subject was instructed to do each movement task for about 10 seconds in one trial. Three trials were collected as the training data. In the real time testing sessions, 10 real-time testing trials were conducted. To evaluate the system performance of real-time intent recognition, we adopted the evaluation criteria as described in our previous study [13]. The testing data were separated into static states and transitional periods. The static state was defined as the state of the subject continuously walking on the same type of terrain (level ground and stair) or performing the same task (standing). A transitional period was the period when subjects switched locomotion modes. The purpose of the UIR system is to predict mode transitions before a critical gait event for safe and smooth switch of prosthesis control mode. In this study the critical timing was defined for each type of transition. For the transitions from standing to locomotion modes (level-ground walking and stair ascent), the critical timing was defined at the beginning of the swing phase (i.e. toe-off). For the transitions from locomotion modes to standing, the critical timing was the

beginning of the double stance phase (i.e. heel contact). The real time performance of our embedded system was evaluated by the following parameters.

Classification Accuracy in the Static States: The classification accuracy in the static state is the percentage of correctly classified observations over the total number of observations in the static states.

The Number of Missed Mode Transitions: For the transitions from standing to locomotion modes, the transition period starts one second before the critical timing, and terminates at the end of the single stance phase after the critical timing; for the transitions from locomotion modes to standing, the transition period includes the full stride cycle prior to the critical timing and the period of one second after the critical timing. A transition is missed if no correct transition decision is made within the defined transition period.

Prediction Time of Mode Transitions: The prediction time of a transition in this experiment is defined as the elapsed time from the moment when the decisions of the classifier changes movement mode to the critical timing for the investigated task transitions.

The overall classification accuracy in the static states across 10 testing trials for classifying level-ground walking, stair ascent and standing was 99.31%. For all the 10 trials, no missed mode transitions were observed within the defined transition period. Table 3.3 lists the average and the standard deviation of the prediction time for four

Table 3.3. Prediction time of mode transitions before critical timing

Transition	**ST→W**	**W→ST**	**ST→SA**	**SA→ST**
Prediction Time (ms)	412.8±76.7	124.39±114.2	549.83±139.2	-104.67±54.1

types of transitions. The results show that there was around 104 ms decision delay for the transitions from stair ascent to standing (SA→ST). This is because the subject could not perform foot-over-foot alternating stair climbing with a passive knee joint. In our experiments, the subject climbed stairs by lifting the sound leg on one step and then pulled up the prosthetic leg on the same step, which produced the same pattern as the mode transition from stair ascent to standing. Therefore the transition SA→ST was only able to be recognized after the subject was standing still. This problem will be eliminated by replacing the passive device with a powered knee in the near future. Wearing the powered knee, the prosthesis user is able to climb stairs foot-over-foot, which provides a very different pattern from the transition SA→ST. For the other three types of transitions (ST→W, W→ST, and ST→SA), the user intent for mode transitions can be accurately predicted 104-549 ms before the critical timing for switching the control of prosthesis. Figure 3.6 shows the real-time system performance

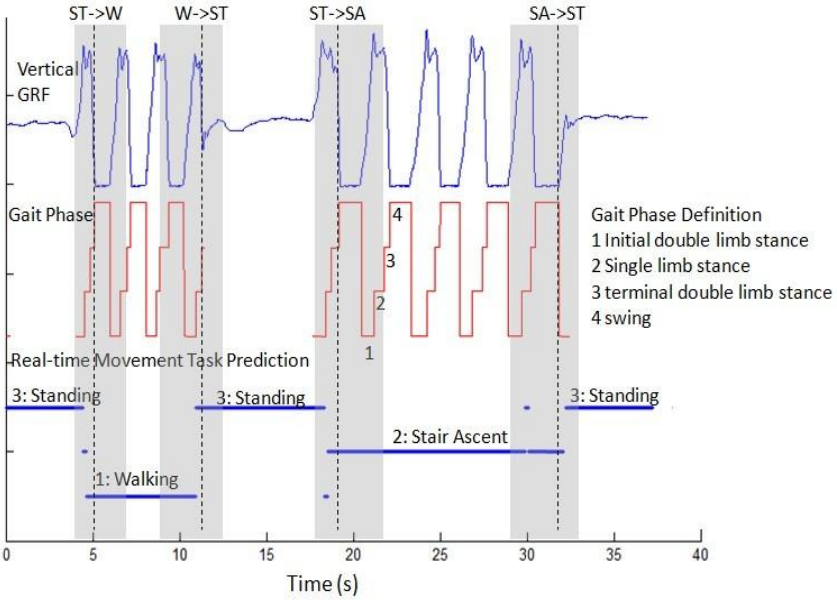

Figure 3.6. Real-time system performance for one representative testing trial. The white area denotes the periods of static states (level-ground walking, stair ascent, and standing); the gray area represents the transitional period; the black vertical dash line indicates the critical timing for each transition.

for one representative testing trial. The white area in Figure 6 denotes the periods of static states (level-ground walking, stair ascent, and standing), the gray area represents the transitional period, and the black vertical dash line indicates the critical timing for each transition. We can see in this trial all the transitions were correctly recognized within the transitional period. No missed classifications occurred in the static states in this trial. The video of our real-time experiments can be found at http://www.youtube.com/watch?v=KNhihjXProU.

3.5 Conclusions

This paper presented the design and implementation of the first complete cyber physical system of neural machine interface for artificial legs. The new CPS implemented both training and testing modules on one single chip, and integrated all the necessary interfaces and control algorithms for identifying the user's intended locomotion modes in real-time. The designed NMI incorporated an MCU for sensing and buffering input EMG signals and mechanical signals, and an FPGA device as the computing engine for fast decoding and pattern recognition. A special parallel processing algorithm for UIR was designed and implemented that realized the neuromuscular-mechanical fusion based PR algorithm coupled with the real-time controlling algorithm on the FPGA. The FPGA implementation of the PR algorithm achieved a speedup of 7X over the Matlab implementation for the training phase, and a speedup of more than 30X for the testing phase with no sacrifice of computation accuracy. The designed NMI prototype was tested on an able-bodied subject for accurately classifying multiple movement tasks (level-ground walking, stair ascent, and standing) in real-time. The results demonstrated the feasibility of a self-contained and

high performance real-time NMI for artificial legs. Our future work includes real-time testing of the designed NMI system on amputee subjects, using the NMI system to control powered prosthetic legs, studying management of power consumption, and increasing the system reliability.

Appendix 3A – Pattern Recognition Using Linear Discriminant Analysis

The principle of the LDA-based PR strategy is to find a linear combination of features which separates multiple locomotion classes $C_g (g \in [1, G])$. G denotes the total number of classes. Suppose μ_g is the mean vector of class C_g and every class shares a common covariance matrix $\sum$, the linear discriminant function is defined as

$$d_{C_g} = \bar{f}^T \sum^{-1} \mu_g - \frac{1}{2} \mu_g{}^T \sum^{-1} \mu_g . \tag{3.1}$$

During the training procedure, $\sum$ and μ_g are estimated based on the feature matrix calculated from the training data. The estimations of $\sum$ and μ_g are expressed as

$$\tilde{\sum} = \frac{1}{G} \sum_{g=1}^{G} \frac{1}{K_g - 1} (F_g - Mi_g)(F_g - Mi_g)^T$$

and

$$\tilde{\mu}_g = \frac{1}{K_g} \sum_{k=1}^{K_g} \bar{f}_{C_g,k}$$

where K_g is the number of analysis windows in class C_g; $\bar{f}_{C_g,k}$ is the k_{th} observed feature vector in class C_g; $F_g = [\bar{f}_{C_g,1}, \bar{f}_{C_g,2}, ..., \bar{f}_{C_g,k}, ..., \bar{f}_{C_g,K_g}]$ is the feature matrix of class C_g; $Mi_g = [\tilde{\mu}_g, \tilde{\mu}_g, ..., \tilde{\mu}_g]$ is the mean matrix that has the same number of

columns as in F_g. The results of the LDA training procedure can be represented by a weight matrix as $W = [\overline{w}_1, \overline{w}_2, ..., \overline{w}_g, ..., \overline{w}_G]$ and a weight vector as $\overline{c} = [c_1, c_2, ..., c_g, ..., c_G]$. Here

$$\overline{w}_g = \tilde{\Sigma}^{-1} \tilde{\mu}_g \tag{3.2}$$

and

$$c_g = -\frac{1}{2} \tilde{\mu}_g^T \tilde{\Sigma}^{-1} \tilde{\mu}_g . \tag{3.3}$$

Therefore (3.1) can be estimated as

$$\tilde{d}_{C_g} = \bar{f}^T \overline{w}_g + c_g . \tag{3.4}$$

The major task of the training procedure is to calculate the mean vector $\tilde{\mu}_g$ for each class, the common covariance matrix $\tilde{\Sigma}$, and its inverse matrix $\tilde{\Sigma}^{-1}$. In practice, matrix inversion is a compute-intensive and time consuming task, which should be avoided if possible. From (3.2) and (3.3), it can be found that $\tilde{\Sigma}^{-1}$ does not appear alone. If $\tilde{\Sigma}^{-1} \tilde{\mu}_g$ can be calculated in an efficient way, W and $\overline{c}$ can be achieved easily. In our implementation, a more efficient algorithm was adopted to solve this problem. First, a Cholesky decomposition is performed as $\tilde{\Sigma} = R^T \times R$, where R is upper triangular. Then $\tilde{\Sigma}^{-1} \tilde{\mu}_g$ can be quickly computed with a forward substitution algorithm for a lower triangular matrix R^T, followed by a back substitution algorithm for an upper triangular matrix R. The condition of a successful Cholesky decomposition is that $\tilde{\Sigma}$ must be symmetric and has real positive diagonal elements,

which can be perfectly satisfied by a covariance matrix. In this way, (3.2) and (3.3) can be reformulated as $\overline{w}_g = R \backslash (R^T \backslash \tilde{\mu}_g)$ and $c_g = -\frac{1}{2} \tilde{\mu}_g{}^T \overline{w}_g$.

During the testing phase, the observed feature vector $\bar{f}$ derived from each analysis window is applied to calculate $\tilde{d}_{C_g}$ in (3.4) for each movement class and is classified into a class $\tilde{C}_g$ that satisfies

$$\tilde{C}_g = \arg\max C_g \{\tilde{d}_{C_g}\}, C_g \in \{C_1, C_2, ..., C_G\}.$$

List of References

[1] Y. Oonishi, S. Oh, and Y. Hori, "A New Control Method for Power-Assisted Wheelchair Based on the Surface Myoelectric Signal," *IEEE Transactions on Industrial Electronics*, vol. 57, pp. 3191-3196, 2010.

[2] L. Tonin, T. Carlson, R. Leeb, and J. R. Millán, "Brain-Controlled Telepresence Robot by Motor-Disabled People," in *33rd Annual International Conference of the IEEE EMBS*, Boston, MA, 2011.

[3] P. Parker and R. Scott, "Myoelectric control of prostheses," *Critical reviews in biomedical engineering*, vol. 13, p. 283, 1986.

[4] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans Biomed Eng*, vol. 50, pp. 848-54, Jul 2003.

[5] C. T. Lin, L. W. Ko, J. C. Chiou, J. R. Duann, R. S. Huang, S. F. Liang, T. W. Chiu, and T. P. Jung, "Noninvasive neural prostheses using mobile and wireless EEG," *Proceedings of the IEEE*, vol. 96, pp. 1167-1183, 2008.

[6] T. Kuiken, "Targeted reinnervation for improved prosthetic function," *Phys Med Rehabil Clin N Am*, vol. 17, pp. 1-13, Feb 2006.

[7] H. Huang, P. Zhou, G. Li, and T. A. Kuiken, "An analysis of EMG electrode configuration for targeted muscle reinnervation based neural machine interface," *IEEE Trans Neural Syst Rehabil Eng*, vol. 16, pp. 37-45, Feb 2008.

[8] H. Huang, T. A. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Trans Biomed Eng*, vol. 56, pp. 65-73, Jan 2009.

[9] F. Zhang, W. DiSanto, J. Ren, Z. Dou, Q. Yang, and H. Huang, "A Novel CPS System for Evaluating a Neural-Machine Interface for Artificial Legs," *Proc. of IEEE/ACM Second International Conference on Cyber Physical Systems,* 2011, pp. 67-76.

[10] X. Zhang, Y. Liu, F. Zhang, J. Ren, Y. L. Sun, Q. Yang, and H. Huang, "On Design and Implementation of Neural-Machine Interface for Artificial Legs," *IEEE Transactions on Industrial Informatics,* vol. 8, pp. 418-429, 2012.

[11] X. Zhang, H. Huang, and Q. Yang, "Design and Implementation of A Special Purpose Embedded System for Neural Machine Interface," *Proc. of IEEE International Conference on Computer Design*, 2010, pp. 166-172.

[12] Altera. Nios II Processor: The World's Most Versatile Embedded Processor. Available: http://www.altera.com/products/ip/processors/nios2/ni2-index.html

[13] H. Huang, F. Zhang, L. Hargrove, Z. Dou, D. Rogers, and K. Englehart, "Continuous Locomotion Mode Identification for Prosthetic Legs based on Neuromuscular-Mechanical Fusion," *IEEE Transactions on Biomedical Engineering*, pp. 1-1, 2011.

MANUSCRIPT 4

An Automatic and User-Driven Training Method for Locomotion Mode Recognition for Artificial Leg Control

by

[1]Xiaorong Zhang, Ding Wang, Qing Yang, and He Huang

is published in *the proceeding of the 34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'12)*,

San Diego, CA, 2012. p. 6116-6120.

[1] Xiaorong Zhang, Ding Wang, Qing Yang, and He Huang are with Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, 02881, Email {zxiaorong, dwang, qyang, huang}@ele.uri.edu.

Abstract

Our previously developed locomotion-mode- recognition (LMR) system has provided a great promise to intuitive control of powered artificial legs. However, the lack of fast, practical training methods is a barrier for clinical use of our LMR system for prosthetic legs. This paper aims to design a new, automatic, and user-driven training method for practical use of LMR system. In this method, a wearable terrain detection interface based on a portable laser distance sensor and an inertial measurement unit (IMU) is applied to detect the terrain change in front of the prosthesis user. The mechanical measurement from the prosthetic pylon is used to detect gait phase. These two streams of information are used to automatically identify the transitions among various locomotion modes, switch the prosthesis control mode, and label the training data with movement class and gait phase in real-time. No external device is required in this training system. In addition, the prosthesis user without assistance from any other experts can do the whole training procedure. The pilot experimental results on an able-bodied subject have demonstrated that our developed new method is accurate and user-friendly, and can significantly simplify the LMR training system and training procedure without sacrificing the system performance. The novel design paves the way for clinical use of our designed LMR system for powered lower limb prosthesis control.

4.1 Introduction

Myoelectric (EMG) pattern recognition (PR) has been widely used for identifying human movement intent to control prostheses [1-4]. The PR strategy usually consists of two phases: a training phase for constructing the parameters of a classifier and a

testing phase for identifying the user intent using the trained classifier. Our previous study has developed a locomotion-mode-recognition (LMR) system for artificial legs based on a phase-dependent PR strategy and neuromuscular-mechanical information fusion [3, 5]. The LMR system has been tested in real-time on both able-bodied subjects and lower limb amputees. The results have shown high accuracies (>98%) in identifying three tested locomotion modes (level-ground walking, stair ascent, and stair descent) and tasks such as sitting and standing [5-6].

One of the challenges for applying the designed LMR system to clinical practice is the lack of practical system training methods. A few PR training methods have been developed for control of upper limb prosthesis, such as screen-guided training (SGT) [7-8], where users perform muscle contractions by following a sequence of visual/audible cues, and prosthesis-guided training (PGT) [9], where the prosthesis itself provides the cues by performing a sequence of preprogrammed motions. However, neither SGT nor PGT can be directly adopted in the training of LMR system for lower limb prostheses because the computer and prosthesis must coordinate with the walking environment to cue the user to perform locomotion mode transitions during training data collection. Currently the training procedure for the LMR system is time consuming and manually conducted by experts. During the training procedure, experts cue the user's actions according to the user's movement status and walking terrain in front of the user, switch the prosthesis control mode before the user steps on another type of terrain, and label the collected training data with movement class manually using an external computer. Such a manual approach significantly challenges the clinical value of LMR because usually the experts are not

available at home.

To address this challenge, this paper aims to design an automatic and user-driven training method for the LMR system. The basic idea is replacing the expert with a smart system to collect and automatically label the training data for PR training. Our design significantly simplifies the training procedure, and can be applied anytime and anywhere, which paves the way for clinical use of the LMR system for powered lower limb prosthesis control.

4.2 Automatic Training Method

The LMR system for artificial legs is based on phase-dependent pattern classification [4-5], which consists of a gait phase detector and multiple sub-classifiers corresponding to each phase. In this study, four gait phases are defined: initial double limb stance (phase 1), single limb stance (phase 2), terminal double limb stance (phase 3), and swing (phase 4) [5]. In the LMR system training, the EMG signals and mechanical forces/moments are the inputs of the LMR system [4] and segmented by overlapped analysis windows. For every analysis window, features of EMG signals and mechanical measurements are extracted from each input channel and concatenated into one feature vector. The feature vector must be labeled with the correct movement class and gait phase to train individual sub-classifiers.

The previous approach labels the training data with locomotion mode (class) and gait phase by an experimenter manually. In this design, we replace the experimenter by a smart system that (1) automatically identifies the transition between locomotion modes based on terrain detection sensors and algorithms, (2) switches the control mode of powered artificial legs, (3) labels the analysis windows with the locomotion

mode (class index) and gait phase, and (4) trains individual sub-classifiers. Our designed system consists of three parts: a gait phase detector for identifying the gait phase of the current analysis window, a terrain detection interface for detecting the terrain in front of the user, and a labeling algorithm to label the mode and gait phase of current data.

To label every analysis window with locomotion mode (class index), it is important to define the timing of mode transition. The purpose of the LMR system is to predict mode transitions before a critical gait event for safe and smooth switch of prosthesis control mode. Our previous study has defined this critical timing for each type of mode transition [4, 6]. In order to allow the LMR system to predict mode transitions before the critical timings, the transition between locomotion modes is defined to be the beginning of the single stance phase (phase 2) immediately prior to the critical timing during the transition [4].

1) Gait Phase Detection: The real-time gait phase detection is implemented by monitoring the vertical ground reaction force (GRF) measured from the 6 DOF load cell. The detailed algorithm can be found in [5].

2) Terrain Detection Interface: Because the sequence of the user's locomotion mode in the training trials is predefined, the goal of the terrain detection interface is not to predict the unknown terrain in front of the subject, but to detect the upcoming terrain change in an appropriate range of distances to help identify the transition between the locomotion modes.

Figure 4.1 shows the sensor setup of the terrain detection interface. A portable laser distance sensor and an inertial measurement unit (IMU) are placed on the

prosthesis user's waist as suggested in [10], because this sensor configuration has been demonstrated to provide stable signals with very small noises and good performance for recognizing terrain types. Before the training procedure starts, a calibration session is conducted first to measure a few parameters for later use in the training process. During calibration, the user walks at a comfortable speed on the level-ground for about 30 seconds. The average vertical distance from the laser sensor to the level ground (H) and the average step length (S_L) of the user are measured.

Three types of terrains have been investigated in this study, including terrains that are above current negotiated terrain (upper terrain), terrains with the same height as the current terrain (level terrain), and terrains that are below the current terrain (lower terrain). The terrain types can be discriminated by a simple decision tree as shown in Figure 4.2. In Figure 4.2, $\tilde{h}(t)$ denotes the estimated height of the terrain in front of

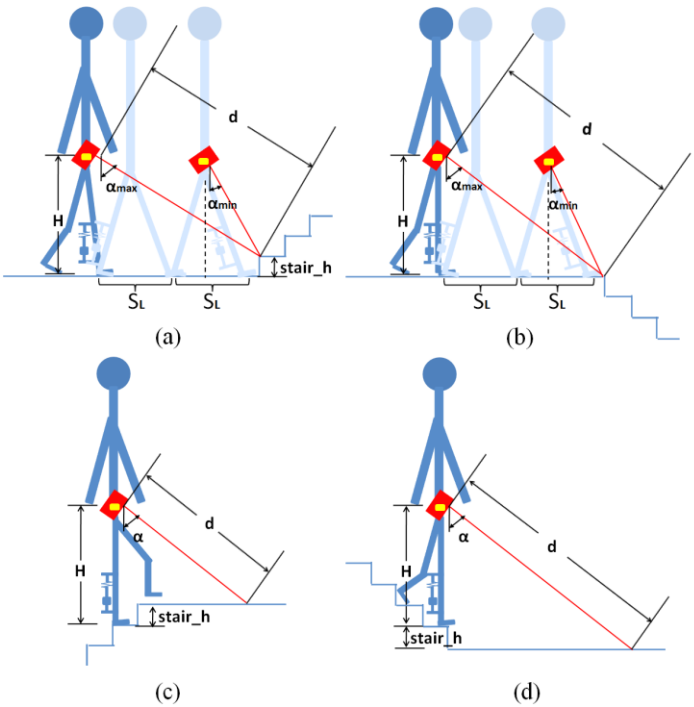

Figure 4.1. Four types of terrain alterations investigated in this study.

the subject, which can be calculated by $\tilde{h}(t) = H - d(t)\cos\alpha(t)$. Here $d(t)$ denotes the distance measured from the laser sensor; $\alpha(t)$ is the angle between the laser beam and the vertical direction, which can be obtained from the IMU; H is the average vertical distance from the laser sensor to the terrain measured in the calibration session. T_{h1} and T_{h2} in Figure 4.2 represent the thresholds that distinguish the three terrain types. To reduce possible miss identifications, only the decisions in phase 1 (i.e. initial double limb stance phase) of each stride cycle are considered for detection of terrain change.

In order to accurately identify the transitions between consecutive movement tasks in real-time, the detection of terrain alteration must happen within the stride cycle immediately prior to the transition point. To meet this requirement the initial angle between the laser beam and the vertical direction (α_{init}) and the thresholds T_{h1} and T_{h2} need to be chosen appropriately. As shown in Figure 4.1(a) and (b) for the terrain alterations from level ground to up/down stair, in order to make sure the subject is within the prior cycle to the transition point when the terrain alteration is detected, by assuming the variation of α during level ground walking is very small, the acceptable range of α_{init} can be estimated by

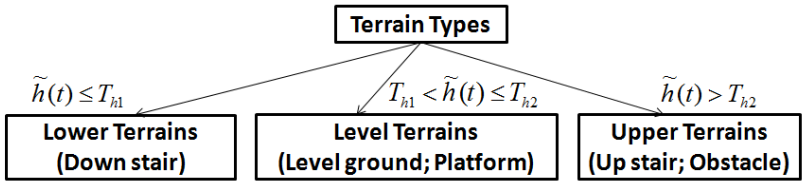

Figure 4.2. The decision tree that discriminates the terrain types.

$$\arctan\left(\frac{0.5\times S_L}{H-T_{h2}}\right) < \alpha_{init} < \arctan\left(\frac{2\times S_L}{H-T_{h2}}\right) \tag{4.1}$$

for Figure 4.1(a), and

$$\arctan\left(\frac{0.5\times S_L}{H}\right) < \alpha_{init} < \arctan\left(\frac{2\times S_L}{H}\right) \tag{4.2}$$

for Figure 4.1(b).

For Figure 4.1(c) and (d), which represent the terrain alterations from up/down stair to level terrain, the terrain alterations must be detected within the last step. To satisfy this condition, T_{h1} and T_{h2} need to be less than the height of one stair step.

3) Labeling Algorithm: The gait phase of every analysis window is directly labeled with the output of the gait phase detector. On the promise of detecting all the terrain alterations in the required ranges by the terrain detection interface, the transition point between locomotion modes is identified as the first analysis window in phase 2 immediately after the expected terrain change is detected. The transition point from standing to locomotion modes can be automatically identified without the information of terrain type, which is the first analysis window immediately after toe-off (the beginning of phase 2). For two consecutive movement modes, the analysis windows before the transition point are labeled with the former movement class, and the windows after the transition point are labeled with the latter movement mode.

4.3 Participant and Experiments

4.3.1 Participant and Measurements

This study was conducted with Institutional Review Board (IRB) approval at the University of Rhode Island and informed consent of subjects. One male able-bodied

subject was recruited. A plastic adaptor was made so that the subject could wear a prosthetic leg on the right side.

Seven surface EMG signals were collected from the thigh muscles on the subject's right leg including adductor magnus (AM), biceps femoris long head (BFL), biceps femoris short head (BFS), rectus femoris (RF), sartorius (SAR), semitendinosus (SEM), and vastus lateralis (VL). The EMG signals were filtered between 20 Hz and 450 Hz with a pass-band gain of 1000. Mechanical ground reaction forces and moments were measured by a 6 degree-of-freedom (DOF) load cell mounted on the prosthetic pylon. A portable optical laser distance sensor and an inertial measurement unit (IMU) were placed on the right waist of the subject. The laser distance sensor could measure a distance ranging from 300 mm to 10000 mm with the resolution of 3 mm. The EMG signals and the mechanical measurements were sampled at 1000 Hz. The signals from the laser sensor and the IMU were sampled at 100 Hz. The input data were synchronized and segmented into a series of 160 ms analysis windows with a 20 ms window increment. For each analysis window, four time-domain (TD) features (mean absolute value, number of zero crossings, waveform length, and number of slope sign changes) were extracted from each EMG channel [11]. For mechanical signals, the maximum, minimum, and mean values calculated from each individual DOF were the features. Linear discriminant analysis (LDA) [12] was used as the classification method for pattern recognition. The system were implemented in Matlab on a PC with 1.6GHz Xeon CPU and 2GB RAM.

4.3.2 Experimental Protocol

In this study, four movement tasks (level-ground walking (W), stair ascent (SA),

stair descent (SD), and standing (ST)), and five mode transitions (ST→W, W→SA, SA→W, W→SD, and SD→W) were investigated. An obstacle course was built in the laboratory, consisting of a level-ground walk way, 5-step stairs with the height of 160 mm for each step, a small flat platform, and an obstacle block (300 mm high and 250 mm wide).

The experiment consisted of three sessions: calibration session, automatic training session, and real-time testing session. Before the training started, the calibration session was conducted to measure the average vertical distance from the laser sensor to the level ground (H) and the average step length (S_L) of the subject. During calibration, the subject walked on the level-ground at a comfortable speed for 30 seconds. H and S_L were measured to be 980 mm and 600 mm, respectively. Because T_{h1} and T_{h2} need to be less than the height of one stair step (160 mm) as explained in Section 4.2, T_{h1} and T_{h2} were set to -120 mm and 120 mm, respectively. From (4.1) and (4.2) derived in Section 4.2, the estimated range of α_{init} was calculated to be (19, 50) degree, and α_{init} was set to 42 degree.

During training, the subject was asked to perform a sequence of predefined movement tasks. The subject began with standing for about four seconds, switched to level-ground walking on the straight walkway, transited to stair ascent, walked on the platform with a 180 degrees turn, transited to stair descent, and switched back to level-ground walking on the walkway, stopped in front of the obstacle, turned 180 degrees, and repeated the previous tasks in the same way for two more times. In this training trial, besides the movement tasks investigated in this study, there were movements not wanted to be included in the training dataset, such as turning in front of the obstacle,

and walking and turning on the platform. These movements were labeled as "not included" (NI) mode.

After training, ten real-time testing trials were conducted to evaluate the performance of the LMR system. Each trial lasted about one minute. All the investigated movement tasks and mode transitions were evaluated in the testing session.

4.4 Results & Discussions

In the training trial, the subject took about 225 seconds to complete all the movement tasks. After the subject finished all the tasks, only 0.11 second was further spent to train the classifiers. All the terrain alterations were accurately identified and all analysis windows were correctly labeled. Figure 4.3 shows the automatic labeling of locomotion modes in part of the training trial. It is observed from the figure that all terrain alterations were recognized at the beginning of phase 1 during the transition cycle, which means the actual terrain changes were detected before phase 1 and within the stride cycle prior to the transition. The transition points between consecutive tasks were accurately identified at the beginning of phase 2 during the transition cycle. All movement tasks were labeled with the correct class modes.

The overall classification accuracy across 10 real-time testing trials was 97.64%. For all the 10 trials, no missed mode transitions were observed. The user intent for mode transitions was accurately predicted 103-653 ms before the critical timing for switching the control of prosthesis. The results indicate that the LMR system using automatic training strategy provides a comparable performance with the system using previous training method.

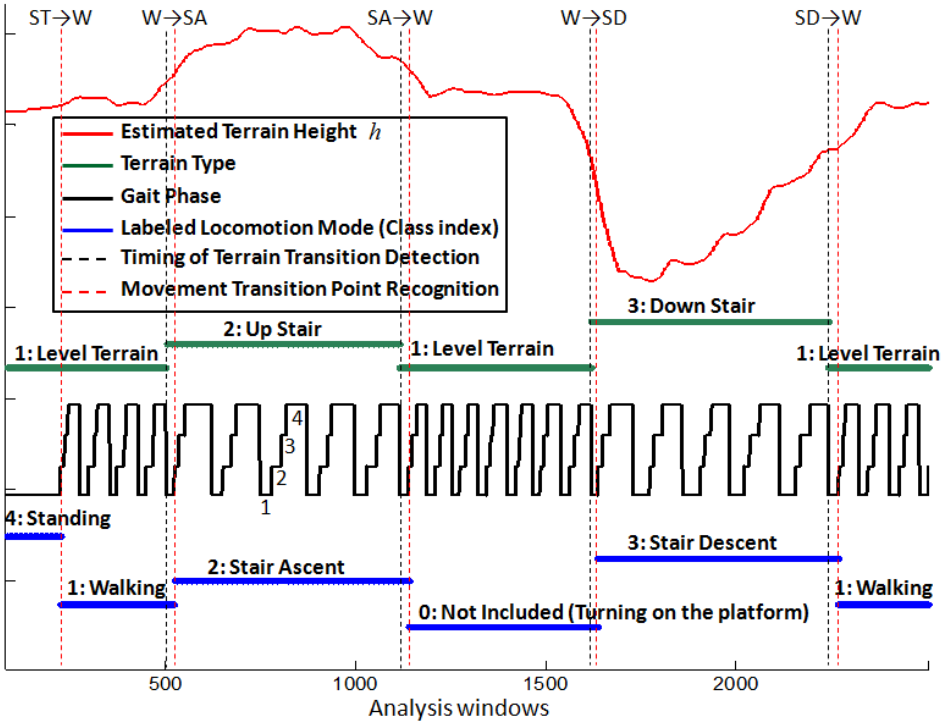

Figure 4.3. Automatic labeling of locomotion modes in part of the training trial.

Table 4.1 summarizes the comparison between our new automatic training method and the previous training method. From the table we can see the new training method can significantly simplify the training procedure and shorten the total training time.

4.5 Conclusions

In this paper, an automatic, user-driven training strategy has been designed and implemented for classifying locomotion modes for control of powered artificial legs. The smart system can automatically identify the locomotion mode transitions based on a terrain detection interface, switch the prosthesis control mode, label the training data with correct mode (i.e. class index) and gait phase in real-time, and train the pattern classifiers in LMR quickly. The preliminary experimental results on an able-bodied subject show that all the analysis windows in the training trial were correctly labeled

Table 4.1. Comparison between the new automatic training method and the previous training method

	New Automatic Training	**Traditional Training**
Connection to external device	A laser distance sensor and an IMU are required, which are both portable, and can be integrated into the prosthesis system in the future	An external computer is required.
Requirement of extra manpower	No.	A professional experimenter is required.
Total training time	30 s calibration time for measuring a few parameters; 225 s for performing movement tasks; 0.11 s for the rest training process;	225 s for performing movement tasks; 24 s for offline processing of training algorithm; At least 10 minutes for interacting with the experimenter, and manual data labeling
Is the system easy to follow?	User-driven: The training can be easily operated by a 'naïve user' unaided. The user only needs to perform all the movement tasks, and the training will be immediately done.	Experimenter driven: The user needs to follow the guidance from the experimenter. The user needs to pause and wait when the experimenter is processing the data.
The way to switch the prosthesis control mode	Automatic switch; Driven by user's motion	Manual switch; Controlled by experimenter

in real-time and the algorithm training process was accomplished immediately after the user completed all the movements. Compared with the system using traditional training strategy, our new training method can significantly simplify the training system and procedure, be easily operated by a naïve user, and shorten the total training time without sacrificing the system performance. These results pave the way for clinically viable LMR for intuitive control of prosthetic legs.

List of References

[1] P. Parker and R. Scott, "Myoelectric control of prostheses," *Critical reviews in biomedical engineering*, vol. 13, p. 283, 1986.

[2] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans Biomed Eng*, vol. 50, pp. 848-54, Jul 2003.

[3] H. Huang, T. A. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Trans Biomed Eng*, vol. 56, pp. 65-73, Jan 2009.

[4] H. Huang, F. Zhang, L. Hargrove, Z. Dou, D. Rogers, and K. Englehart, "Continuous Locomotion Mode Identification for Prosthetic Legs based on Neuromuscular-Mechanical Fusion," *IEEE Trans Biomed Eng*, vol. 58, pp. 2867-75, 2011.

[5] F. Zhang, W. DiSanto, J. Ren, Z. Dou, Q. Yang, and H. Huang, "A novel CPS system for evaluating a neural-machine interface for artificial legs," *Proc. of IEEE/ACM Second International Conference on Cyber Physical Systems*, 2011, pp. 67-76.

[6] F. Zhang, Z. Dou, M. Nunnery, and H. Huang, "Real-time Implementation of an Intent Recognition System for Artificial Legs," in *33rd Annual International Conference of the IEEE EMBS*, Boston, MA, USA, 2011.

[7] E. Scheme and K. Englehart, "A flexible user interface for rapid prototyping of advanced real-time myoelectric control schemes," *Proc. of Myoelectric Controls Symposium*, 2008, pp. 150-55.

[8] T. A. Kuiken, G. Li, B. A. Lock, R. D. Lipschutz, L. A. Miller, K. A. Stubblefield, and K. B. Englehart, "Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms," *Jama*, vol. 301, pp. 619-28, Feb 11 2009.

[9] B. A. Lock, A. M. Simon, K. Stubblefield, and L. J. Hargrove, "Prosthesis-Guided Training For Practical Use Of Pattern Recognition Control Of Prostheses," *Proc. of Myoelectric Controls Symposium*, Fredericton, New Brunswick, Canada, Aug, 2011.

[10] F. Zhang, Z. Fang, M. Liu, and H. Huang, "Preliminary design of a terrain recognition system," in *33rd Annual International Conference of the IEEE EMBS*, Boston, MA, USA, 2011.

[11] B. Hudgins, P. Parker, and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Trans Biomed Eng*, vol. 40, pp. 82-94, Jan 1993.

[12] H. Huang, P. Zhou, G. Li, and T. A. Kuiken, "An analysis of EMG electrode configuration for targeted muscle reinnervation based neural machine interface," *IEEE Trans Neural Syst Rehabil Eng*, vol. 16, pp. 37-45, Feb 2008.

MANUSCRIPT 5

Real-Time Implementation of a Self-Recovery EMG Pattern Recognition Interface for Artificial Arms

by

[1]Xiaorong Zhang, He Huang, and Qing Yang

is submitted to *the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'13)*,

Osaka, Japan, 2013.

[1] Xiaorong Zhang, He Huang, and Qing Yang are with Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, 02881, Email {zxiaorong, huang, qyang}@ele.uri.edu.

Abstract

EMG pattern classification has been widely studied for decoding user intent for intuitive prosthesis control. However, EMG signals can be easily contaminated by noise and disturbances, which may degrade the classification performance. This study aims to design a real-time self-recovery EMG pattern classification interface to provide reliable user intent recognition for multifunctional prosthetic arm control. A novel self-recovery module consisting of multiple sensor fault detectors and a fast LDA classifier retraining strategy has been developed to immediately recover the classification performance from signal disturbances. The self-recovery EMG pattern recognition (PR) system has been implemented on an embedded system as a working prototype. Experimental evaluation has been performed on an able-bodied subject in real-time to classify three arm movements while signal disturbances were manually introduced. The results of this study may propel the clinical use of EMG PR for multifunctional prosthetic arm control.

5.1 Introduction

Electromyographic signal (EMG) pattern recognition (PR) is a widely used method for classifying user intent for neural control of artificial limbs [1-3]. However, unreliability of surface EMG recordings over time is a challenge for applying the EMG pattern recognition controlled prostheses for clinical practice. Motion artifacts, environmental noises, sensor location shifts, user fatigue, and other conditions may all cause changes in the EMG characteristics and thus lead to inaccurate identification of user intent and threaten the prosthesis control reliability and user safety[4-5].

Several strategies have been developed to address this challenge in order to make

artificial limb control based on EMG PR clinically viable. Sensinger et al. [5] employed adaptive pattern classifier to cope with variations in EMG signals for reliable EMG PR. Tkach et al. [6] investigated different EMG features and suggested several time-domain features that were resilience to EMG signal change caused by muscle fatigue and exerted force levels. Hargrove et al. [7] suggested a new EMG PR training procedure in order to accommodate EMG electrode shift during prosthesis use.

Our research group developed a unique, reliable EMG pattern recognition interface, consisting of sensor fault detectors and a self-recovery mechanism. The sensor fault detectors monitor the recordings from individual EMG electrodes; the self-recovery mechanism will remove the faulty EMG signals from the PR algorithm to recover the classification accuracy [8-10]. It was observed that the EMG classification performance was not significantly affected by the removal of one or two EMG signals from redundant EMG recordings [2, 8]. Our new EMG-PR interface could salvage system performance by up to 20% increased classification accuracy when one or more EMG signals were disturbed [8].

Despite the promise of our design concept showed in our previous study, the algorithm development and validation were tested offline. In order to implement this concept in real-time, especially in a wearable embedded system, several challenges still exist. First, the recovery strategy involves retraining of the pattern classifier. Currently this procedure involves reorganization of training feature matrix, computation of parameters in the pattern classifiers, and reorganization of testing feature vectors. Whether or not the embedded system can handle this procedure

quickly for each decision-making is unknown. Secondly, since more components are included in the EMG PR algorithm, communication among components and precise timing control is crucial. Finally, a compact integration of all the components in an embedded computer is required. The system needs to provide necessary interfaces for data collection, adequate computing power for real-time decision making, efficient memory management, and low power consumption. All these challenges have never been explored.

This paper presents the first real-time self-recovery EMG pattern recognition interface for artificial arms. A novel self-recovery scheme with a fast and efficient retraining algorithm based on linear discriminant analysis (LDA) has been developed. The self-recovery EMG pattern recognition system was implemented on an embedded computer system as a working prototype. The prototype was preliminarily evaluated on an able-bodied subject in real-time in classifying three arm movements while motion artifacts were manually introduced by randomly tapping the EMG electrodes. The results of this study may propel the clinical use of EMG PR for multifunctional prosthetic arm control.

5.2 Methods

5.2.1 System Structure

The overall structure of the self-recovery EMG pattern recognition interface is shown in Figure 5.1. The system seamlessly integrates EMG pattern recognition with the self-recovery module. Multiple channels of EMG signals segmented by overlapped sliding analysis windows are the system inputs. In each window, four time-domain (TD) features (mean absolute value, number of zero crossings, waveform length, and

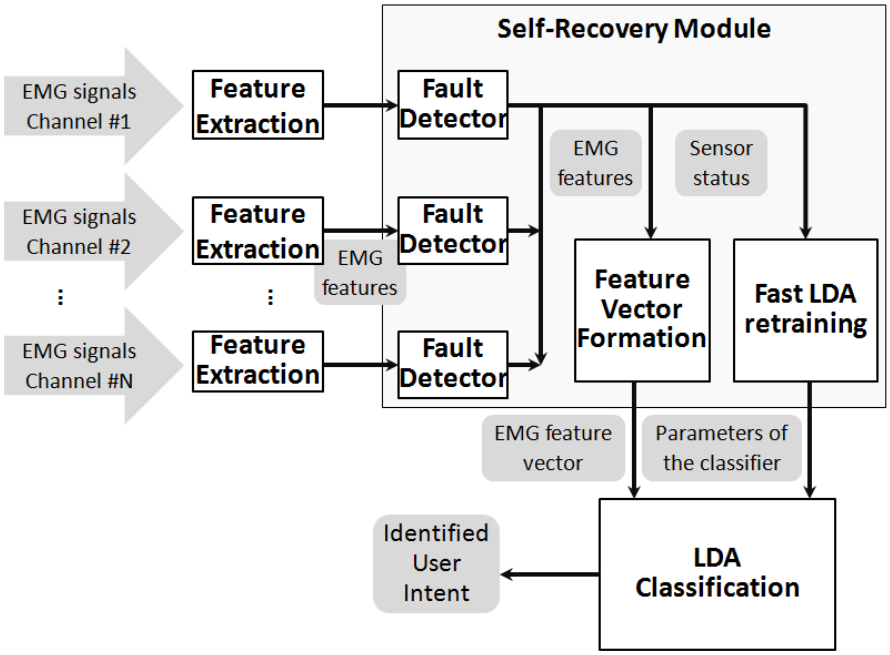

Figure 5.1. System structure of the self-recovery EMG sensing interface for LDA-based pattern recognition.

number of slope sign changes [11]) of the EMG signals are extracted from each input channel and fed to the self-recovery module. The sensor fault detectors closely monitor the key features of each EMG signal to detect disturbances. Based on the detection results, the EMG features extracted from 'normal' channels are concatenated into a feature vector as the input for pattern classification. If no disturbance is detected, the feature vector is directly sent to the classifier generated from the original training data. If one or more signals are determined as 'abnormal', the fast LDA retraining process is triggered and the reduced feature vector is fed to the new classifier for pattern recognition.

5.2.2 Fast LDA-based Retraining Algorithm

Previously the lack of a fast and efficient retraining algorithm was the most critical challenge to the design of a real-time self-recovery EMG PR interface. If the

retraining process cannot be accomplished in a short period of time, the signal disturbances may impair the classification performance and even harm the prostheses users' safety. Linear discriminant analysis (LDA) is a widely used method for EMG pattern recognition [1, 10-11]. By examining the details of the LDA algorithm, we developed a fast and memory efficient LDA retraining algorithm by making the most efficient use of existing information.

The principle of the LDA-based PR strategy is to find a linear combination of features which separates multiple classes $C_g (g \in [1,G])$. Here G denotes the total number of studied classes. Suppose $\bar{f}$ is the feature vector in one analysis window, μ_g is the mean vector of class C_g and every class shares a common covariance matrix $\sum$, the LDA function is defined as $d_{C_g} = \bar{f}^T \sum^{-1} \mu_g - \frac{1}{2} \mu_g{}^T \sum^{-1} \mu_g$.

During the training procedure, $\sum$ and μ_g are estimated based on the feature matrix calculated from the training data. The estimations of $\sum$ and μ_g are expressed as

$$\tilde{\sum} = \frac{1}{G} \sum_{g=1}^{G} \frac{1}{K_g - 1} (F_g - Mi_g)(F_g - Mi_g)^T$$

and

$$\tilde{\mu}_g = \frac{1}{K_g} \sum_{k=1}^{K_g} \bar{f}_{C_g,k}$$

where K_g is the number of analysis windows in class C_g; $\bar{f}_{C_g,k}$ is the k_{th} observed feature vector in class C_g; $F_g = [\bar{f}_{C_g,1}, \bar{f}_{C_g,2}, ..., \bar{f}_{C_g,k}, ..., \bar{f}_{C_g,K_g}]$ is the feature matrix of class C_g; $Mi_g = [\tilde{\mu}_g, \tilde{\mu}_g, ..., \tilde{\mu}_g]$ is the mean matrix which has the same dimension as

F_g. In a feature vector $\bar{f}_{C_g,k} = [f_1, f_2, ..., f_n, ..., f_N]^T$, N is the total number of EMG input channels and f_n denotes the four EMG features extracted from the n_{th} channel.

In the previous retraining strategy [8], after the initial training process is done, the original EMG feature matrix is stored in the memory for later use in the retraining process. During the retraining procedure, for each class, a new EMG feature matrix F_g' is reorganized by removing the feature rows corresponding to the disturbed channels from F_g. The mean vector of each class $\tilde{\mu}_g'$ and the new common covariance matrix $\tilde{\Sigma}'$ are then recalculated based on F_g'. Our experimental analysis has shown that the calculation of $\tilde{\Sigma}'$ is the most computational intensive task in the retraining procedure, which accounts for more than 90% of the total processing time. This is because for each class, a large amount of analysis windows are collected as the training data. The number of columns in F_g' may vary from several hundreds to a few thousands, which leads to intensive numerical operations in calculating $\tilde{\Sigma}'$.

Fortunately, after closely analyzing the details of the LDA training algorithm, we have found that the calculation of $\tilde{\Sigma}'$ and $\tilde{\mu}_g'$ can be avoided in a smart way. The trick is, instead of the large feature matrix F_g, only $\tilde{\mu}_g$ and $\tilde{\Sigma}$ are stored in the memory after the initial training process is finished. $\tilde{\Sigma}'$ and $\tilde{\mu}_g'$ can be easily retrieved from $\tilde{\Sigma}$ and $\tilde{\mu}_g$. Figure 5.2 shows an example of the retrieving process if a single EMG channel is detected to be 'abnormal'. Assume there are totally 6 EMG channels. Each element in the mean vector is calculated by averaging one specific feature row in F_g. Therefore $\tilde{\mu}_g'$ can be obtained by taking off the four elements that are associated with

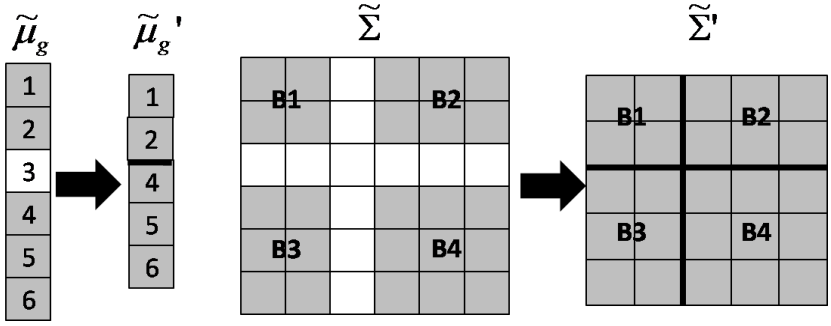

Figure 5.2. An example of retrieving $\tilde{\Sigma}'$ and $\tilde{\mu}_g'$ from $\tilde{\Sigma}$ and $\tilde{\mu}_g$ when a single EMG channel is disturbed. The white blocks represent the elements associated with the disturbed channel.

the disturbed EMG channel from $\tilde{\mu}_g$. $\tilde{\Sigma}'$ is constructed by removing the corresponding rows and columns associated with the disturbed channel from $\tilde{\Sigma}$ and then merging the remaining four small matrices ($B1$, $B2$, $B3$, and $B4$ in Figure 5.2). If multiple EMG signals are disturbed, $\tilde{\Sigma}'$ and $\tilde{\mu}_g'$ can be obtained by doing the retrieving process repeatedly. Compared with the previous retraining algorithm which requires intensive numerical operations and a large memory space, the new strategy dramatically accelerates the retraining speed and is much more memory efficient.

5.2.3 Sensor Fault Detection

To detect individual EMG sensor abnormalities, various signal processing methods have been applied to sensor fault detection [8-10]. A detector based on Bayesian decision rule [8] has been proposed for accurately detecting three types of simulated distortions including EMG signal drift and saturation, additional noise in the signal, and variation of EMG magnitude. An abnormality detector using Cumulative Sum (CUSUM) algorithm [9] has been developed to closely monitor the changes of EMG features for detecting sudden changes or gradual changes in EMG signals.

In this study, the CUSUM detector is adopted in our implementation because of

its computational efficiency for real-time processing, its high accuracy, and low false alarm rate in detecting motion artifacts [9-10]. Two EMG features including mean absolute value and number of zero crossings are monitored to recognize abnormal changes. Detailed algorithms of the CUSUM detector can be found in [9].

5.2.4 Real-Time Embedded System Implementation

A preliminary prototype of the self-recovery EMG pattern recognition system was implemented on Gumstix Overo Air, an ARM Cortex-A8 OMAP3503 based computer-on-module (COM), and RoboVero, an expansion board with an ARM Cortex-M3 microcontroller and eight 12-bit analog-to-digital converters (Fig. 3). The Overo COM communicates with the RoboVero expansion board via two 70-pin connectors as shown in Figure 5.3. The system implementation consists of two parts: the microcontroller on the RoboVero expansion board for data sampling and dispatching, and the Cortex-A8 processor on the Overo COM for EMG pattern recognition.

5.2.5 Experimental Protocol

This study was conducted with Institutional Review Board (IRB) approval at the

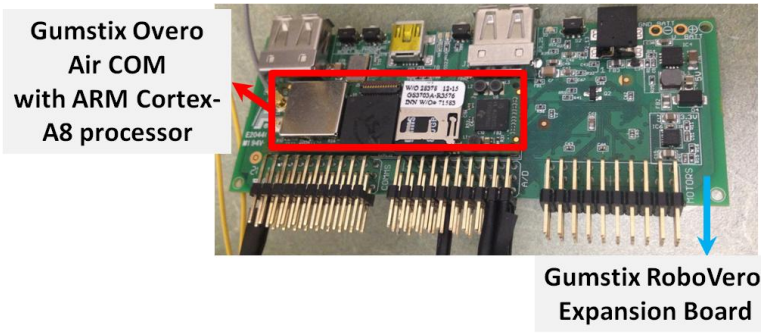

Figure 5.3. The prototype based on Gumstix Overo Air COM and RoboVero expansion board.

University of Rhode Island and informed consent of subject. One male able-bodied subject was recruited. Four surface EMG electrodes (MA-420-002, Motion Lab System Inc.) were placed around the subject's right forearm. An MA-300 EMG system collected four channels of EMG signals. The analog EMG signals were digitally sampled at the rate of 1000 Hz by the Gumstix RoboVero expansion board. The sampled data were segmented into overlapped analysis windows with 160 ms length and 20 ms increment, resulting in a new decision every 20 ms. Three motion classes (Elbow Flexion, Elbow Extension, and No Movement) were investigated in this experiment. The experiment consisted of two sessions: training session, and testing session.

The training session was conducted first to collect the training data and build the original classifier. The subject was instructed to perform one movement for about 4 seconds in one trial. For each movement task, three separate trials were collected. After the training process was done, the parameters of the generated classifier, as well as the mean vector for each class and the common covariance matrix were saved in the memory for later use in the testing session.

In the real-time testing session, for each movement task, the subject performed the movement for about 4 seconds in four separate trials. Totally 12 testing trials were conducted. In every trial, motion artifacts were manually introduced by randomly tapping the EMG electrodes with roughly equal strength. In the preliminary experiment, we only tapped one electrode at a time. To better evaluate the performance of our self-recovery module, two types of classification decisions with and without the self-recovery module were compared in every analysis window.

In addition, an offline evaluation was conducted to compare the performance between our fast LDA retraining algorithm and the previous retraining strategy [8, 10] by processing the same dataset collected in the real-time testing session.

5.3 Results & Discussions

5.3.1 Performance of the Retraining Algorithm

Table 5.1 summarizes the comparison between our new fast LDA retraining algorithm and the previous retraining algorithm. From the table we can see the new retraining algorithm was two orders of magnitude (118 times) faster than the previous retraining strategy and meanwhile only consumed less than 1% of the memory usage of the old strategy. Furthermore, our fast retraining algorithm only took less than 1 ms to generate the new classifier. This result makes it possible for the system to extract EMG features, detect signal disturbances, retrain the classifier, perform pattern recognition, and produce a decision seamlessly in a sequence within the duration of

Table 5.1. Comparison between the new retraining method and the previous retraining method

	New Fast Retraining	**Previous Retraining**
Processing time	0.55 ms (2307 windows, 3 classes, 4 channels)	65 ms (2307 windows, 3 classes, 4 channels)
Speedup	118	1
Memory Usage (2307 windows, 3 classes, 4 channels, 4 features per channel)	$\tilde{\mu}_g$:(4x4)x4 bytes=64 bytes; $\tilde{\Sigma}$: (4x4)x(4x4)x4 bytes =1024 bytes; Total: 64x3+1024 = 1216 bytes=1.2 Kbytes	Total size of the feature matrix: (4x4)x2307x4 bytes = 147648 bytes = 144.2 Kbytes
Meet real-time constraints?	Yes.	No.

one window increment (i.e. 20 ms). This new design and implementation clearly demonstrated the feasibility of a self-recovery strategy that is truly 'imperceptible' to users.

5.3.2 System Performance in Real-Time

In the 12 real-time testing trials, totally 48 motion artifacts were introduced, among which 43 were recognized by the CUSUM detector and 20 caused miss classifications if our self-recovery was not used. All the disturbances that led to classification errors were successfully detected. The undetected disturbances were those with either small amplitude or short duration, which did not affect the classification performance. Without the self-recovery module, there were 277 miss classifications observed among 5993 decisions. All these errors were caused by motion artifacts. Our self-recovery module eliminated 259 of them, resulting in a 93.5% recovery rate.

Figure 5.4 shows the real-time system performance of some representative testing trials. The blue line at the bottom demonstrates one channel of the EMG signals which was randomly disturbed by motion artifacts. The black line above is the detection results of the CUSUM detector. As seen in the figure, the CUSUM detector accurately recognized all five motion artifacts. The classification decisions without self-recovery are displayed by the red line. The green line denotes the recovered decisions. The three gray ellipses in the figure mark three typical cases in the experiment. Case *A* represents a situation in which the self-recovery module successfully eliminates the classification error caused by motion artifacts. This is also the most common case. *B* is a case in which the sensor fault detector identifies the disturbance but the retrained

classifier still provides an incorrect decision. This may be because the disturbed EMG

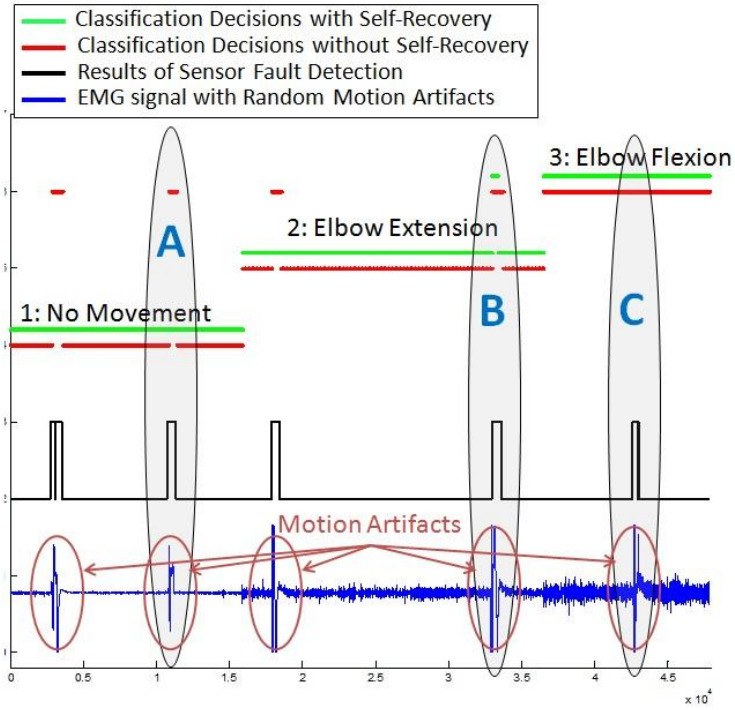

Figure 5.4. Real-time system performance of some representative testing trials.

signal is critical to the recognition of this motion. Another case *C* is a situation where the disturbance does not affect the classification decision.

The results of the experiment have shown the promise of a robust, reliable, and efficient real-time EMG pattern recognition interface for artificial arms.

5.4 Conclusion

This paper presented a real-time self-recovery EMG pattern recognition interface for artificial arms. The system seamlessly integrated EMG pattern recognition with a self-recovery module that could detect signal disturbances, retrain the classifier, and perform reliable pattern classification in real-time. A novel fast and efficient LDA-based retraining algorithm was developed and demonstrated the ability to immediately recover the classification performance from motion artifacts. The self-recovery EMG

pattern recognition system was implemented on an embedded computer system as a working prototype. The preliminary experimental evaluation on an able-bodied subject showed that our system could maintain high accuracy in classifying three arm movements while motion artifacts were manually introduced. The self-recovery module was able to eliminate 93.5% of the miss classifications caused by motion artifacts. These results have demonstrated the feasibility of a clinically viable EMG PR interface for multifunctional prosthetic arm control.

List of References

[1] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans Biomed Eng*, 2003. 50(7): pp. 848-854.

[2] H. Huang, T. A. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Trans Biomed Eng*, 2009. 56(1): pp. 65-73.

[3] X. Zhang, H. Huang, and Q. Yang, "Implementing an FPGA System for Real-Time Intent Recognition for Prosthetic Legs," *Proc. of Design Automation Conference*, 2012. pp. 169-75.

[4] P. Parker, K. Englehart, and B. Hudgins, "Myoelectric signal processing for control of powered limb prostheses," *Journal of electromyography and kinesiology*, 2006. 16(6): pp. 541.

[5] J. W. Sensinger, B. A. Lock, and T. A. Kuiken, "Adaptive pattern recognition of myoelectric signals: exploration of conceptual framework and practical algorithms," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2009. 17(3): pp. 270-278.

[6] D. Tkach, H. Huang, and T. A. Kuiken, "Study of stability of time-domain features for electromyographic pattern recognition," *J Neuroeng Rehabil*, 2010. 7: p. 21.

[7] L. Hargrove, K. Englehart, and B. Hudgins, "A training strategy to reduce classification degradation due to electrode displacements in pattern recognition based myoelectric control," *Biomedical Signal Processing and Control*, 2008. 3(2): pp. 175-180.

[8] H. Huang, F. Zhang, Y. L. Sun, and H. He, "Design of a robust EMG sensing interface for pattern classification," *J Neural Eng*, 2010. 7(5): pp. 056005.

[9] Y. Liu, F. Zhang, Y. Sun, and H. Huang, "Trust sensor interface for improving reliability of EMG-based user intent recognition," in *33rd Annual International Conference of the IEEE EMBS*, Boston, MA, USA, 2011.

[10] X. Zhang, Y. Liu, F. Zhang, J. Ren, Y. L. Sun, Q. Yang, and H. Huang, "On Design and Implementation of Neural-Machine Interface for Artificial Legs," *IEEE Transactions on Industrial Informatics*, 2012. 8(2): pp. 418-429.

[11] B. Hudgins, P. Parker, and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Trans Biomed Eng*, 1993. 40(1): p. 82-94.