INTUITIVE HUMAN ROBOT INTERFACES FOR UPPER LIMB PROSTHETICS

by

OGUZ YETKIN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2016

To my daughter Cansu

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr.Dan Popa as well as Dr. George Alexandrakis, Dr. Young-tae Kim, Dr. Rita Patterson, Dr. Nicoleta Bugnariu, Dr. Digant Dave, and Dr. Vassilis Athitsos for their guidance and valuable constructive criticism.

I would also like to extend my appreciation to the members of the Next Generation Systems lab, especially Joe Sanford, Sumit Kumar Das, Fahad Mirza, J Paul Carpenter, Roopak Karulkar, Joshua Baptist, Ruoshi Zhang, Drew Waller, Dr. Indika Vijayasinghe, and Brandon Young. I am also grateful for the many hours spent by my students Simranjit Ahluwalia, Dinithi Silva, Isioma Kasi-Okonye, Nitzajaret Elizondo, Ta Er Al Kuor, Nawal Aditya and Laura Jesness. In addition, the feedback from the graduate student writing group at the University of Louisville has been quite valuable in the preparation of this manuscript.

I am also grateful for my friends Kristi Wallace, Loan Bui, Stacy Wylie, Stephen Wylie and Sanny Garrett as well as my friends and collaborators at the Dallas Makerspace who have spent countless hours helping me debug and construct hardware including Ebony Jackson, Chad Bonner, William Kitchen, Karen Kitchen, Asenath Kitchen, Brian Terry, Rachael Volker, Romeo Espana, Ashley Newland, Otto Wagner, and many others.

Finally, I would like to express my deep gratitude to my mother Dr. Zerrin Yetkin, my father Dr. Mustafa Enis Yetkin, my wife Hope Yetkin and my daughter Cansu Deniz Yetkin who have all sacrificed many years away from me and without whom I could not have complete my studies.

July 8, 2016

ABSTRACT

INTUITIVE HUMAN ROBOT INTERFACES FOR UPPER LIMB PROSTHETICS

OGUZ YETKIN, Ph.D.

The University of Texas at Arlington, 2016

Supervising Professor: Dan O. Popa

Modern robotic prosthetic devices for upper limb amputees promise to alleviate an important disability, but are underutilized due to the inability to properly control them. Specifically, the devices afford more degrees of freedom (DOFs) than are controllable by easily decoded biological signals. These devices, such as the DEKA arm, can have as many as 18 DOFs, although six is a more typical number (control of each finger plus thumb rotation). Unfortunately, the use of these devices remains limited by the ability of users to simultaneously control more than one degree of freedom at a time with commercially deployed technology.

Control of robotic prosthetic devices is typically achieved through electromyogram (EMG) signals read from the residual limb. While several groups have reported being able to use multiple EMG sensors to classify the user intent from residual muscle activity, such systems have not proven robust enough to translate to clinical use and are not intuitive.

In the first part of this research, the prosthetic control problem is re-framed as a Human Robot Interface problem, developing and clinically evaluating several robotic interface methods which can eliminate or complement the use of EMG signals while allowing the user to quickly achieve more grasping patterns, thus allowing the use of all the DOFs available in the prosthetic device. Three healthy limb based methods have been developed and evaluated, including: 1) the use of the healthy hand to teleoperate the proshtetic device via a Mirroring Glove, 2) the use of the healthy hand to issue pre-programmed commands to the prosthetic device via a Gesture Glove and 3) the use of the healthy hand with extremely light fingernail worn devices to issue commands to the prosthetic device.

In the second part of this research, a field-deployable and easy way of training a multiple input based EMG classifier is presented and extended to using Force Myography (FMG) data fused with EMG data.

Overall, a number of different experiments were conducted with a total of 20 human subjects, including 2 amputees, and the following conclusions were reached: 1) Healthy limb based prosthetic device control can match the performance speed of EMG based control with very little training 2) Gesture based control of the healthy limb is faster than mirrored teleoperation except in the case of tasks which are mirrored by their nature 3) Bilateral hand movements combined with kinematic tracking of the healthy limb can be utilized to train a Force Myography (FMG) based classifier as well as an EMG based classifier, and that the combination of the two modalities hold promise to make a readily deployable multi-DOF EMG/FMG classifier system a reality.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

ABSTRACT vi

LIST OF ILLUSTRATIONS xiii

LIST OF TABLES xix

Chapter Page

1. INTRODUCTION 1

1.1 Problem Statement 2

1.2 Research Aims 3

1.3 Human Robot Interface Considerations 4

1.3.1 Goals of UI Design 4

1.4 Experimentation During Research 6

1.5 EMG and FMG classification experiments 8

1.6 Publications 9

1.6.1 Journal Articles and Patents 10

1.6.2 Conference Papers 10

1.7 Posters and Presentations 10

1.8 Research Contributions 11

1.8.1 Aim 1 11

1.8.2 Aim 2 12

1.9 Thesis Organization 13

2. BACKGROUND ON PROSTHETIC CONTROL 14

2.1 Historical Perspective on Prosthetic Device Control 16

2.2 State of the Art in Prosthetic Device Control 18
2.3 Healthy Limb Based Control 22
2.3.1 Existing Systems 23
2.4 Enabling Technologies for Healthy Limb Based Control 23
2.4.1 State of the Art in Hand and Gesture Tracking Technology . . 24
2.4.2 Gesture Recognition Systems Developed 30
2.5 Electromyography 32
2.5.1 Electromyography Signal Acquisition and Processing 32
2.5.2 EMG Based Classification Algorithms 33
2.5.3 The Need for a Better Classifier Training Scheme 34
2.5.4 Existing Systems of Relevance 34
2.5.5 Use of Bilateral Movements for Training 34
2.5.6 Optimal Electrode Placement 35
2.5.7 Dimensionality Reduction 36
2.6 Force Myography 37
3. HEALTHY LIMB AND GLOVE BASED INTERFACES 39
3.1 Introduction 39
3.1.1 System Architecture 39
3.1.2 Healthy Limb Adapter 40
3.2 Mirroring Glove 43
3.2.1 Mirroring Glove Construction 44
3.2.2 Mirroring Glove Software 46
3.2.3 Mirroring Glove Testing 47
3.3 Gesture Glove 47
3.3.1 Gesture Glove Construction 48
3.3.2 Gesture Glove Operation 50

3.4 Experiments Performed 56
3.5 General Experimental Protocols 56
3.5.1 Experiment 1: Towel Folding Test 56
3.5.2 Experiment 2: Box and Blocks Test 56
3.5.3 Experiment 3: Blanket Folding Test 57
3.5.4 Experiment 4: Bottle Carrying Test 57
3.5.5 Experiment 5: Jar Test 58
3.6 Experimental Results 58
3.6.1 Towel and Blanket Folding Tests 58
3.6.2 Gesture Glove vs. EMG in Blanket Folding Test 58
3.6.3 Box and Blocks Test 60
3.6.4 Bottle Carrying Test 63
3.7 Discussion 63
3.7.1 Developed Systems in the Context of UI Design 65
3.8 Conclusion 66
3.9 Future Work 66
4. DEVELOPMENT AND TESTING OF A LIGHTWEIGHT FINGERNAIL WORN SENSOR 68
4.1 Motivation 68
4.2 Physical Design of Current System 70
4.3 Detection Methods 77
4.3.1 Time Based Detection 78
4.3.2 Light Interrogator Based Detection 79
4.3.3 FFT Based Detection 80
4.4 Experiments Performed 83
4.4.1 Box and Blocks Test 84

4.4.2 Jar Test 85
4.4.3 Blanket Folding Test 85
4.4.4 Tray Carrying Test 86
4.4.5 Tissue Characterization Experiment 86
4.4.6 Gesture Performance Experiment 87
4.4.7 Single Digit Response Time Experiment 88
4.4.8 FFT Based Detection through Two Fingers 89
4.5 Experimental Results 89
4.5.1 Box and Blocks Experiment Results 90
4.5.2 Jar Test Results 91
4.5.3 Blanket Folding Test Results 92
4.5.4 Tray Carrying Test Results 93
4.5.5 Tissue Characterization Results 94
4.5.6 Gesture Performance Experiment 96
4.5.7 FFT Based Detection Through Two Fingers 98
4.6 Discussion 99
4.6.1 Gesture Performance Experiment 99
4.6.2 FFT Based Detection through Two Fingers 100
4.6.3 Results of Single Digit Response Time 100
4.7 Conclusion and Future Work 103
4.7.1 Future Work 104
5. USING BILATERAL MOVEMENTS TO TRAIN EMG AND FMG CLASSIFIERS 107
5.1 Problem Statement 107
5.2 Experiments Performed 108
5.2.1 Data Acquisition 108

5.2.2 Multichannel EMG recordings on Healthy Subjects 114
5.2.3 EMG Recording on Amputee 122
5.2.4 Force Myography Recording 125
5.2.5 FMG Results 126
5.3 Classification Results from FMG 128
5.3.1 FMG Data Import and Preprocessing 128
5.3.2 Obtaining User Intent from Glove Data 129
5.4 Discussion 134
5.5 Future Work 135
5.5.1 EMG/FMG based Classifier Training 135
5.5.2 Kinematic Tracking based on Fingernail Device 136
6. CONCLUSION 137
6.1 Discussion 137
6.2 Future Work 138
6.2.1 Envisioned System 139
Appendix
A. SOURCE CODE FOR EMBEDDED UNIX SERVER AND MIRRORING GLOVE SYSTEM 140
B. SOURCE CODE FOR GESTURE GLOVE SYSTEM 153
C. SOURCE CODE FOR FINGERNAIL SENSOR SYSTEM 159
D. SOURCE CODE EMG AND FMG CLASSIFICATION 167
REFERENCES 181
BIOGRAPHICAL STATEMENT 194

LIST OF ILLUSTRATIONS

Figure Page

2.1 The DEKA Arm. Image from [25]. 15

2.2 Body powered prosthetic device. Image from [27]. 16

2.3 Adaptive Prosthetics. Left and Top: Custom prosthetics for recreational activities. Images from [29]. Bottom Right: Prosthetic attachment for gardening. Image from [30]. 17

2.4 Several State of the Art prosthetic hands. Top Left: *bebionic* hand, image from [31]. Top Right: Otto Bock Michelangelo Hand, image from [32]. Bottom Left: Vincent Hand, image from [35]. Bottom Right: TouchBionics iLimb Quantum, image from [33]. 19

2.5 Touch Bionics i-mo interface for associating grip patterns with accelerometer based gestures. Image from [34]. 20

2.6 EMG Classification from Targeted Muscle Reinnervation (TMR). Image from [39]. 21

2.7 CyberGlove hand tracking system with CyberGrasp exoskeleton. Image reproduced from [45]. 25

2.8 Wrist mounted camera system for gesture tracking from Kim et al. [49]. Image reproduced from [49]. 27

2.9 The Myo armband is a gesture input device utilizing eight EMG sensors as well as accelerometers. 28

2.10 The *gest* glove. 29

2.11 Light Transmission through the hand. Reproduced from [55]. 31

2.12 Sensor Placement from Maier et al. [76] 35
2.13 Multiple EMG sensor array from Sueaseenak et al. [9]. 37
3.1 System Architecture. A: Glove Based Control Architecture. Either glove is worn on the sound hand. B: Left: The Gesture Glove and two gestures, mapped to two different poses. B: Right: The Mirroring Glove and two gestures being mirrored on the prosthetic limb. 40
3.2 Healthy Limb Adapter. 42
3.3 Updated version of the healthy limb adapter featuring a handhold and a softer OrthoFlex shell. 43
3.4 Some typical Mirroring Glove uses. 44
3.5 Mirroring Glove with lock button. 45
3.6 Steps involved in making a sewable sensor holder. 46
3.7 Raw data from flexion sensors on one finger on repeated opening and closing of the hand. 48
3.8 Gesture Glove with some possible actions. 49
3.9 Raw data from conductive thread pads. The time series shows consecutive touch events for the index, middle, ring, and little fingers. 52
3.10 Values from A/D converter filtered with a moving average filter with a window size of n = 50, showing four consecutive touch events (index, middle, ring, and little finger). 53
3.11 The touch events from above displayed as standard deviations from the running average at each point. Note that each touch value goes above 2.5 SD from mean in this case. 54

3.12 Touch events, O, identified after the application of threshold Tstd. The system outputs a value of Oi = 100 (arbitrarily chosen) if the Activation Statistic, Ai, is greater than the standard deviation threshold Tstd. (Note: this figure was obtained from a different run of the experiment than the previous figures). 55
3.13 Results of the towel folding task. 59
3.14 Result of the blanket folding task, where the advantage of using the Gesture Glove based prosthetic system can be seen. The subjects were asked to fold a large (243cm x 243cm) blanket. Using the Gesture Glove system is significantly faster than using a single hand ($p < 0.05$). . . . 60
3.15 Box and Blocks Test, instance 1 (comparison with iPhone). 61
3.16 Box and Blocks Test, instance 2 (comparison with EMG). 62
3.17 Bottle Carrying Test. Left: Time taken for successful attempts. Right: Average number of failures before the achievement of a successful attempt for each control scheme. 63
4.1 Basic circuit for proof of concept implementation of Fingernail Device. 71
4.2 Left: Sensor holder design and implementation Right: Photodiode based detector circuit. 72
4.3 Bipolar Junction Transistor and OpAmp based circuit to amplify light through two fingers. One flashing LED is shown on the right. Circuit diagram from Mr. Brandon Young. 73
4.4 Transimpedance amplifier used with a 16 Bit Analog to Digital converter device which as its own programmable gain. Circuit diagram from Mr. Simranjit Ahluwalia. 74
4.5 Proof of Concept system with a wrist-worn microcontroller (top) and three pinch gestures (bottom). 75

4.6 Light transmission through connected fingers when one finger is illuminated. 76
4.7 Light detected on a fingernail after traversing through two fingers in physical contact (top), and light detected on the same fingernail after traversing through two fingers with a small air gap between them. . . . 77
4.8 Light Interrogator Based Detection. 80
4.9 FFT based detection simulation with and without noise. Left: output of frequency_detector() for simulated data, no noise. F1= 40 Hz and F2= 20 Hz. Right: output of frequency_detector() for simulated data, + noise 60 Hz + random noise. F1= 40 Hz and F2= 20Hz. 81
4.10 Ground Truth compared with Detected Result. 83
4.11 Two separate gestures performed by the Fingernail Device resulting in two different prosthetic device configurations. 84
4.12 Gesture Prompt Protocol. Action 0 (no LED): No gesture. Action 1 (Yellow LED): Touch pad of index finger to thumbnail. Action 2 (Blue LED): Touch pad of a middle finger to thumbnail. A total of 10 touch events were recorded along with the time data and a value indicating which gesture was being prompted. 87
4.13 Performance Experiment Setup. 88
4.14 Box and Blocks Experiment Results for Fingernail Device, EMG, and the other devices discussed earlier. 91
4.15 Jar Test Results for Fingernail Device, EMG, and the other devices discussed earlier. 92
4.16 Blanket Folding Experiment results for Fingernail Device, EMG, and the other devices discussed earlier. 93

4.17 Tray Carrying Experiment results for Fingernail Device, healthy hand, and the other devices discussed earlier. 94
4.18 Tissue Characterization Experiment Results. 96
4.19 Performance Experiment Details. The red trace indicates the prompt given by the LED prompter illustrated above. The blue trace indicates the gesture as detected by the gesture detection system. 97
4.20 Gesture Performance Experiment Results. 98
4.21 FFT Experiment Results. 99
4.22 Single digit response time. 102
4.23 Single digit response time, sorted from fastest to slowest. Each bar represents 120 trials. 103
5.1 Custom EMG Circuit Used to Acquire the Data. The first stage on the left is the instrument amplifier (AD623). The amplifier was designed and built by Mr. Chad Bonner and Mr. William Kitchen. Diagram used with permission. 108
5.2 Data acquired from sensors on the anterior and posterior of the forearm during the same movement. 109
5.3 Training set for flexion and extension and predicted output using both the training set (left bottom) and the validation set (right bottom). . . 111
5.4 Experimental setup to acquire data to distinguish between pinkie and index finger flexion. 112
5.5 Preliminary data from the two finger discrimination experiment. . . . 113
5.6 Output of neural network for discrimination between index and pinkie fingers. 114
5.7 Custom EMG Shield for Arduino Mega Microcontroller Board. 115
5.8 Recording setup for healthy subject. 116

5.9 Overall Neural Network training scheme. 117
5.10 Operation of the Neural Network once trained. 118
5.11 The Glove Preprocessor Neural Network (GPNN) trained on LED prompter data. 119
5.12 LED Prompter Device. 120
5.13 Preliminary digit position prediction from EMG data. 121
5.14 EMG Data from Fist Clenching Experiment. 122
5.15 EMG Setup on amputee. Left: affected hand with type 5 symbrachydactyly. Middle: Setup with Bagnolli 16 System and Prompter LEDs Right: Setup with Arduino, Tracking Glove, and Prompting LEDs. . 123
5.16 Data from amputee. Top: Ground truth based on LED prompts. Middle: Glove Data. Bottom: EMG channels. 124
5.17 Force Myography Setup. 126
5.18 Eight channels of FMG. 127
5.19 A single isolated channel of FMG. 128
5.20 Imported Data from FMG and Glove Experiment. 129
5.21 Classification of glove data by GPNN for creating ground truth for subsequent steps. 130
5.22 Visual representation of the neural network which uses glove data (GPNN) in order to create ground truth for subsequent steps of the classification experiment. 131
5.23 FMG based finger movement prediction by UIPNN (trained using ground truth). 132
5.24 FMG based finger movement prediction by UIPNN (trained using glove data processed by GPNN). 133

LIST OF TABLES

Table Page

4.1 Fingernail Device performance Compared with Other Devices. The experiment was performed on two healthy volunteers and one amputee. All values are given in seconds ± standard deviation. 90

4.2 Average Single Digit Response Times in Seconds. The experiment was performed on two male and two female right handed volunteers ages 22-47 with no manual dexterity problems. 100

CHAPTER 1

INTRODUCTION

Every year, approximately 25,000 new upper limb amputations occur in the US alone [1]. While prosthetic devices have been in use for centuries, it is only in the last few decades that robotic prosthetic devices have become available. In this work we define robotic prosthetics to mean those devices that are either powered, motorized, which are also sensorized and include advancements in battery and microcontroller technologies. Examples of these new prosthetic devices, such as the DEKA arm, can have as many as 18 degrees of freedom (DOFs) [2], approaching the 22 DOFs afforded by the human arm. These devices are typically controlled by Electromyography but are limited to the control of a single DOF at a time.

In a related area, there has been extensive research in Brain Computer Interfaces (BCIs) for use in prosthetics [3], including the reading and decoding of signals directly from the surface of the brain via electrocorticography (ECoG) to control the arm [4] [5] and the fingers [6]. Furthermore, research in the area of Peripheral Neural Interface (PNI) devices has attempted to interface prosthetic devices directly to the peripheral nerves controlling the muscles of the arm [7]. These systems vary in their degree of invasiveness, and hold great promise for the future. They are out of the scope of the present work, which will focus solely on non-invasive robotic prosthetic interfaces.

1.1 Problem Statement

EMG based prosthetic control has some serious limitations. Users typically reposition their powered prosthetic device by controlling a single degree of freedom (such as joint, finger, or wrist rotator) at a time. In an EMG based hierarchical menu system, a user decides which degree of freedom to control, indicates preference by issuing an EMG based command (such as co-contracting two muscles or contracting a muscle for at longer time than normal), controls that degree of freedom with an EMG command (by contracting one of the muscles being read by the system), and repeats the process to switch to the next degree of freedom in a similar manner until the task is completed. A similar scheme is used to select which grasp pattern (such as a lateral grasp, pinch grasp, etc) to control via EMG. Cycling through different degrees of freedom or grasp patterns in this manner is both time consuming and difficult to learn for the user. Some manufacturers [8] provide a smartphone app connected wirelessly to the device in order to aid in this grasp selection process.

The ideal system would, of course, be one in which each degree of freedom can be controlled in a natural way. Simultaneous control of multiple Degrees of Freedom (DOFs) using EMG signals is an active area of research which, to our knowledge at the time of this writing, has not yielded clinically deployable systems. Although several groups [9] [10] [11] have reported progress in the use of multiple EMG sensors to classify the user intent from residual muscle activity, such systems have not proven robust enough to translate to clinical use and are not intuitive. The field of robotic prosthetics is, therefore, confronted with an interesting problem: The number of DOFs available on a robotic prosthetic device has far surpassed the ability to simultaneously control them with readily available biological signals.

1.2 Research Aims

This research explores two separate approaches to solving the multi-DOF prosthetic control problem and allowing the user to exploit all the available DOFs in a robotic prosthetic arm in order to regain the functionality that a natural hand would normally provide.

In the first part of this research (Aim 1), the prosthetic control problem is reframed as a Human Robot Interface problem. Three novel robotic interface methods (with prototype devices) are presented and tested: A device which can directly control the prosthetic hand via mirroring the movements of the intact hand (the Mirroring Glove), a lightweight glove which allows the user to issue discrete commands to the system (the Gesture Glove), and an extremely lightweight device free of haptic encumbrances (the Fingernail Device). These methods and devices can either eliminate or complement the use of EMG signals while allowing the user to quickly achieve more grasping patterns, thus using all the DOFs available in the prosthetic device.

In the second part of this research (Aim 2), one of the devices created for Aim 1 (the Mirroring Glove) is used as an input method to train an intent classifier system. In addition to EMG, this system utilizes pressure sensitive robotic skin for Force Myography (FMG). This classifier training system works by allowing the user to perform bilateral mirrored movements with the intact hand (while it is being tracked by the Mirroring Glove or a similar device) and the amputated limb. EMG and FMG information from the amputated limb is also tracked and used to train the classifier system. The purpose of this system is to render training/calibration of such EMG/FMG classifier easy enough to be performed by the user each time the device is worn. Such systems generally perform well in the lab and perform poorly once

the user moves the arm out of an ideal resting position. Training and deployment considerations for this system will be addressed.

Once fully developed and clinically deployed, the methods of Aim 1 and Aim 2 should prove to be complementary.

Part of our work in this research focuses on the classification of surface EMG (sEMG) and Force Myography (FMG) signals, so that they can be used to control robotic prosthetic devices. It is possible to obtain more accurate EMG signals via slightly more invasive methods such as needle electrodes [12] or implanted wireless EMG electrodes [13, 14, 15, 16]. While the use of these devices are out of the scope of the present work, the same methods proposed here can easily be extended to needle electrode based EMG or implanted EMG devices.

1.3 Human Robot Interface Considerations

Prosthetic device control is a Human Robot Interface (HRI) problem which falls into the general category of User Interface (UI) and User Experience (UX) problems. Such interfaces have been widely studied in the context of Virtual Reality (VR) and in the context of the Human Computer Interface (HCI) systems (such as keyboards, mice, menus, and touch screen based interfaces) which have become part of daily life.

1.3.1 Goals of UI Design

The main goals of UI design are to communicate human intent to a computer or mechanical system as quickly and efficiently as possible with a minimal amount of error. Schneiderman [17] proposes the following measurable goals for UI design:

1. Time to learn

2. Speed of performance
3. Rate of errors
4. Retention over time
5. Subjective satisfaction

(list reproduced from [17], p15).

These goals may involve tradeoffs. For example, some systems designed for experts in a field may tolerate a high time to learn as an acceptable trade-off for increased speed of performance (the UNIX vi editor [18] is a famous example of a system designed with the goal of performance speed instead of minimizing the time to learn. The editor allows user to move around the screen using the "h,j,k, and l" keys and has a separate mode to edit text or navigate on the screen which is not intuitive to any novice user. Expert users, however, are able to perform very fast file manipulations with this editor). On the other hand, a system controlling a nuclear power reactor, missile launcher, or medical equipment needs to focus on reduction or elimination of errors, for which performance or training time can be sacrificed.

A related concept in the UI design field is that of "friction" [19] which is any impediment the user experiences in the flow of work. While a "low friction" system is generally desired (in which the user can accomplish a desired event without interrupting the flow of work), some have argued that friction can at times be helpful in the form of "microboundaries" [20] (an example of this would be a confirmation dialog or a menu in a UI which is intentionally difficult to get to because it can cause harmful operation, such as an option to erase all data on a system).

A major design consideration of the systems designed for Aim 1 was that the devices would be very quick to learn, have high performance speed, be low in error, and that they would be subjectively satisfying to the user.

1.4 Experimentation During Research

During the course of this research, I have undertaken considerable prototyping of prosthetic interfaces, and experimentation with human subjects to assess their performance. A total number of 20 subjects were recruited under the approved Institutional Review Board (IRB) protocol number 2012-0728.5, including 2 amputees who visited our lab. They tested various aspects of the following prototype interfaces:

1. Three versions of the healthy limb adapter, which is a device created for this work allowing non-amputees to wear an operate a robotic prosthetic device over their existing arm. Each version has been a refinement on the former version, making the device more ergonomic and lighter. The latest version features a hand-hold and a custom OrthoFlex soft silicone shell as well as a fastening system.
2. Three versions of the Mirroring Glove, which is a device allowing the teleoperation of the robotic prosthetic hand by mirroring the motions of the intact hand. The latest version has more refined user-adjustable velcro-based holders and does not require any amplification to read each flexion sensor.
3. The Gesture Glove, which is a lightweight device allowing the user to command the robotic prosthetic hand by making simple hand gestures. The device was created using an Arduino microcontroller, a fingerless weightlifting glove, and embroidered sensor pads along with a custom signal processing program to recognize gestures running on the microcontroller device.

4. The Fingernail Device, which is an extremely lightweight device allowing the user to command the robotic prosthetic hand by making simple finger gestures. This device uses LED based emitters mounted on the fingernails and a thumbnail mounted sensor to detect touch events.

During the course of research, we experimented with these prototypes by collecting user data and refining our hardware and software as follows:

1. Interfacing the gesture input system to the robotic hand by the creation of a Controller Area Network (CAN) based controller architecture running on a Raspberry Pi embedded computer to translate Mirrored Glove, Gesture Glove, or Fingernail Device commands from the Arduino based microcontroller board to the TouchBionics robo-limb prosthetic device.
2. Extensive experimentation on the hardware and software of the Fingernail Device, including:
   (a) Testing of various LED/photodiode/phototransistor combinations
   (b) Identification of of an appropriate 16 bit Analog to Digital (ADC) controller board
   (c) Testing and identification of several light amplification circuits
   (d) Creation and testing of a device based on LED timing
   (e) Creation and testing of a device based on a "light interrogator" algorithm
   (f) Prototyping of a device based on running an FFT on the received light signal
   (g) Filing of a provisional patent on the fingernail device.
3. Timed testing of the iPhone based interface, Mirrored Glove, and Gesture Glove on the Box and Blocks test.

4. Timed testing of using two healthy hands, a single healthy hand, an unpowered prosthetic hand, and powered prosthetic hand with Mirrored Glove and Gesture Glove on the following activities of daily living:
   (a) Folding a small towel
   (b) Folding a large towel
   (c) Folding a large blanket
   (d) Carrying a laundry basket.
5. Timed testing of using two healthy hands, a single healthy hand, and powered prosthetic hand with Mirrored Glove and Gesture Glove on various activities of daily living (ADLs) including carrying a tray of bottles, opening a jar and putting objects in it, and folding towels and blankets of various sizes.
6. Validation of Mirroring Glove and Gesture Glove data against timing data acquired from an amputee volunteer performing tasks with her own EMG based device
7. Qualitative validation of proposed interfaces with two amputees, who were instructed in how to use the Gesture Glove, Mirroring Glove, and Fingernail Device.

1.5 EMG and FMG classification experiments

During the course of the research for Aim 2, many pieces of new hardware and software have been created in order to simultaneously acquire pressure sensor, EMG sensor and hand tracking data as well as "ground truth" data recorded from the LED array based prompter. This has included:

1. Creation of an LED array based prompter device and scheme in which the user is prompted automatically to perform certain movements and in which the

prompts are logged as "ground truth" for use in visual comparisons and the training of various neural network classification algorithms.

2. Creation of a custom single channel EMG circuit and preliminary experiments with "multi-channel-placement" in which the single channel sensors are moved around and the data acquired with repeated motions.
3. Creation of custom multi-channel EMG data acquisition board for use with an Arduino Mega device in order to create a portable system.
4. Creation of software to synchronize data being acquired from The Mirroring Glove, Force Sensors, and EMG sensors
5. Creation of MATLAB based program utilizing neural networks to simplify acquired Mirroring Glove based data for use as an input in further Neural Network training to classify EMG and FMG data.
6. Creation of a 16 channel Force Myography (FMG) sensor array.
7. Creation of a plastic-molded forearm adapter in order to facilitate acquisition of FMG sensor data.
8. Acquisition of FMG data with glove data from healthy volunteers.
9. Acquisition of EMG data, glove data, and prompter data from an amputee subject.
10. Acquisition of EMG data, glove data, and prompter data from healthy subjects.

1.6 Publications

The research conducted has resulted in several publications, one of which is still under review at the time of writing:

1.6.1 Journal Articles and Patents

1. Yetkin, O., Sanford, J., Mirza, F., Karulkar, R., Das, S. K., & Popa, D. O. (2015). Control of a Powered Prosthetic Hand Via a Tracked Glove. Journal of Medical Devices, 9(2), 020920.
2. Yetkin, Oguz et al. "A Novel Prosthetic Interface based on Hand and Gesture Tracking." Prosthetics & Orthotics International. (Under Review).
3. Yetkin, Oguz et al. (Provisional Patent). "System and Methods for Controlling Devices" US 62/323,592.

1.6.2 Conference Papers

1. Yetkin, O., Wallace, K., Sanford, J. D., & Popa, D. O. (2015, June). Control of a powered prosthetic device via a pinch gesture interface. In SPIE Sensing Technology+ Applications (pp. 94940I-94940I). International Society for Optics and Photonics.
2. Yetkin, O., Ahluwalia, S., Silva, D., Kasi-Okonye, I., Volker, R., Baptist, J. R., & Popa, D. O. (2016, May). An extremely lightweight fingernail worn prosthetic interface device. In SPIE Commercial+ Scientific Sensing and Imaging (pp. 98590J-98590J). International Society for Optics and Photonics.
3. Sanford, J., Yetkin, O., Cremer, S., & Popa, D. O. (2015, July). A novel EMG-free prosthetic interface system using intra-socket force measurement and pinch gestures. In Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments (p. 72). ACM.

1.7 Posters and Presentations

Several oral and poster presentations have also been given on the research and related topics, which are listed below:

1. "Blink/EEG Based Control of Multi DOF Robotic Actuator". Poster. BMES 2014, San Antonio, TX.
2. "Control of a Powered Prosthetic Device via a Pinch Gesture Interface" IEEE Medical Devices Symposium, Dallas TX.
3. Control of a powered prosthetic device via a pinch gesture interface. In SPIE Sensing Technology+ Applications. Baltimore, MD 2015.
4. "Control of a Powered Prosthetic Hand Via a Tracked Glove", UMN Design of Medical Devices Conference, St. Paul, MN April 2015.
5. "A novel EMG-free prosthetic interface system using intra-socket force measurement and pinch gestures". International Conference on PErvasive Technologies Related to Assistive Environments. Corfu, Greece July 2015.
6. Novel Human Robot Interface Methodologies for Prosthetic Device Control. ACES Symposium, University of Texas at Arlington. Spring 2016.
7. An extremely lightweight fingernail worn prosthetic interface device. In SPIE Sensing Technology+ Applications. Baltimore, MD 2016.

1.8 Research Contributions

The research contributions of this thesis associated with my two aims are as follows:

1.8.1 Aim 1

1. Invented, prototyped, and verified a novel system for fingernail based pinch gesture tracking.
2. Proposed, prototyped, and validated the use of hand tracking (via a Mirroring Glove) for prosthetic device control.

3. Proposed, prototyped, and validated the use of pinch gestures for prosthetic device control.
4. Created a new type of fingerless glove for pinch gesture tracking (Gesture Glove)
5. Established a subclass of tasks for which powered prosthetic devices are useful
6. Gathered timing data on different types of fingernail based gestures. Determined that some of the fastest gestures are based on using the non-dominant hand.

1.8.2 Aim 2

The goal of Aim 2 is to create and prototype a field-deployable system which fuses data from both EMG and FMG sensors, and in which the classifier can be trained in the field by the user by utilizing one of the hand-tracking systems discussed earlier by performing the same bi-lateral motion.

This work synthesizes several of the concepts discussed above: The final envisioned system is a dense array of EMG and FMG sensors which are used to train an Artificial Neural Network (ANN) based classifier in a manner which can be easily deployed to a clinical setting.

The research contribution of Aim 2 is the validation of using FMG data (both by itself and fused with EMG) for bilateral movement based neural network classifier training. To the best of the author's knowledge, bilateral training on FMG data or fused FMG and EMG data has not been performed or validated.

The final envisioned system encompasses both the healthy limb based modalities discussed earlier and an EMG/FMG training system in which the healthy limb

based control devices can be used to turn various features of the prosthetic control system on and off.

1.9 Thesis Organization

Chapter 1 (this chapter) introduces the aims and contributions of the research work. Chapter 2 introduces the area of prosthetic device control, presents a brief survey of the state of the art in the topic, and the various fields this thesis draws on: hand and gesture tracking, light transmission through tissue, electromyography (EMG), EMG classification algorithms, and force myography (FMG). Chapter 3 discusses the design of human experiments performed to validate healthy limb based control, including the use of a Mirroring Glove to teleoperate the robotic prosthetic limb with the intact hand, and the use of a discrete gesture input device to command the robotic prosthetic limb with pre-programmed movements. Chapter 4 delves into technical detail about the design, construction, and validation of the novel fingernail based tracking system (Fingernail Device) presented in this thesis. Chapter 5 presents the systems constructed and experiments performed for Aim 2: EMG and FMG based signal classification using tracked kinematic hand data as neural network training input. Chapter 6 concludes this thesis and outlines a future vision of how the components of the thesis can and should be brought together for a new type of prosthetic control system utilizing both EMG, FMG, and healthy limb tracking in order to allow the prosthetic device user the greatest amount of functionality that the current state of technology will allow.

CHAPTER 2

BACKGROUND ON PROSTHETIC CONTROL

The primary goal of upper limb prosthetic devices is to restore functionality and independence to the user. A secondary goal is to provide a device which resembles the missing limb for cosmetic purposes [21]. The ideal prosthetic device would look, act, and feel like the actual missing limb. This deep-seated desire in the field is famously illustrated in the popular movie franchise Star Wars [22] in which one of the characters (Luke Skywalker) obtains a replacement hand after an amputation incident and continues his daily activities without giving the artificial limb a second thought after a minimal amount of calibration. This fictional system appears to have inspired many actual systems while perhaps setting an unrealistic public expectation of what is currently possible in the state of the art at the time of this writing. For example, Dean Kamen's DEKA [23] Arm System (Fig. 2.1) is referred to in many popular media outlets as "The Luke Arm" [24].

Figure 2.1 The DEKA Arm. Image from [25].

Unrealistic public expectations notwithstanding, great advances have been made in both robotic prosthetics and prosthetic control in the last several decades. This section discusses the history of prosthetic devices, summarizes the state of the art in prosthetic device control, and presents an overview of several more specific fields and technologies on which the systems described in the thesis are based. Specifically: Healthy Limb Based Control and Challenges Presented; Hand Tracking Technologies; Gesture Based Device Control; Light Transmission and Detection Through Tissue; Electromyography (EMG); Force Myography (FMG); Artificial Neural Network Based Classification and EMG Classification Based Control of Prosthetic Devices.

2.1 Historical Perspective on Prosthetic Device Control

While some form of prosthetic device has been around for centuries, the prosthetic grabber entered into popular use around the civil war. The most popular device, to this day, remains the cable based prosthetic grabber (patented by William Selpho in 1857 [26]) which allows the user, with the help of a system of shoulder mounted cables, to open a spring loaded grabber, grab an object, and carry or manipulate it. This device is popular due to the fact that it is low cost, lightweight, relatively easy to operate, and does not have the issue of running out of batteries. An example of this type of system is illustrated in Fig. 2.2 (reproduced from [27]).

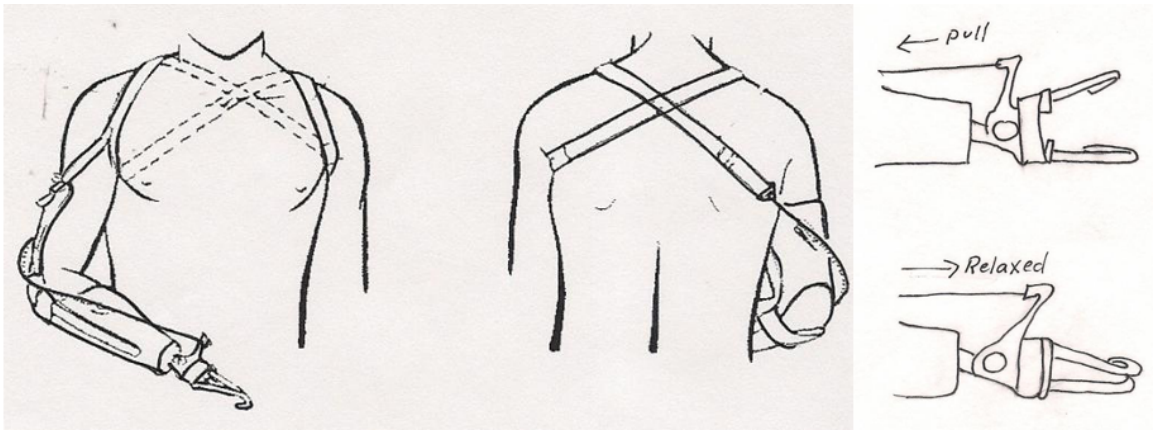

Figure 2.2 Body powered prosthetic device. Image from [27].

In addition to the passive grabber, some amputees choose to wear single-purpose interchangeable implements on their prosthetic socket (such as gardening tools, rock climbing tools, etc.) since they would have normally been holding this tool in their hand to start with. These are generally known as adaptive prosthetics and are available for various sporting [28] [29], gardening [30], or other purposes.

Figure 2.3 Adaptive Prosthetics. Left and Top: Custom prosthetics for recreational activities. Images from [29]. Bottom Right: Prosthetic attachment for gardening. Image from [30].

Unfortunately, both the passive grabbers and the single purpose prosthetics present challenges. The passive grabber does require the user to perform some unnatural body movements, which can be awkward. Single purpose prosthetics also require time to put on and interchange. Human civilization and the tools which it has developed for millennia presuppose the use of the hand. Many users also prefer a hand-shaped prosthetic for cosmetic purposes. Fortunately, size reductions and improvements in EMG, microcontroller, and battery technologies have made Robotic Prosthetic Devices possible in the last three decades.

2.2 State of the Art in Prosthetic Device Control

Currently available robotic prosthetic systems typically rely on one or two Electromyography sensors and allow the user to open or close the robotic hand by issuing one of two commands. Some examples of the currently available hands can be seen in Fig. 2.4. Each of these devices feature the ability to move the fingers independently and have tried to improve on the use of EMG technology alone. The *bebionic* hand [31] features 14 built-in grip positions and pressure sensors in order detect slipping objects and grab them tighter. Otto Bock's Michelangelo prosthetic hand [32] features individually controllable fingers as well as finger abduction/adduction in order to help the user hold small objects such as credit cards between the fingers. The Bionics iLimb Quantum [33] device features controllable fingers, an opposable thumb, as well as various control methods such as an iPhone app [8], actuation via "proximity control grip chips" (small devices which can be affixed to objects which, in turn, cause the hand to automatically assume a pre-set grip pattern), and even control via accelerometer based gestures [34], as seen in Fig. 2.5, as well as EMG based control.

Figure 2.4 Several State of the Art prosthetic hands. Top Left: *bebionic* hand, image from [31]. Top Right: Otto Bock Michelangelo Hand, image from [32]. Bottom Left: Vincent Hand, image from [35]. Bottom Right: TouchBionics iLimb Quantum, image from [33].

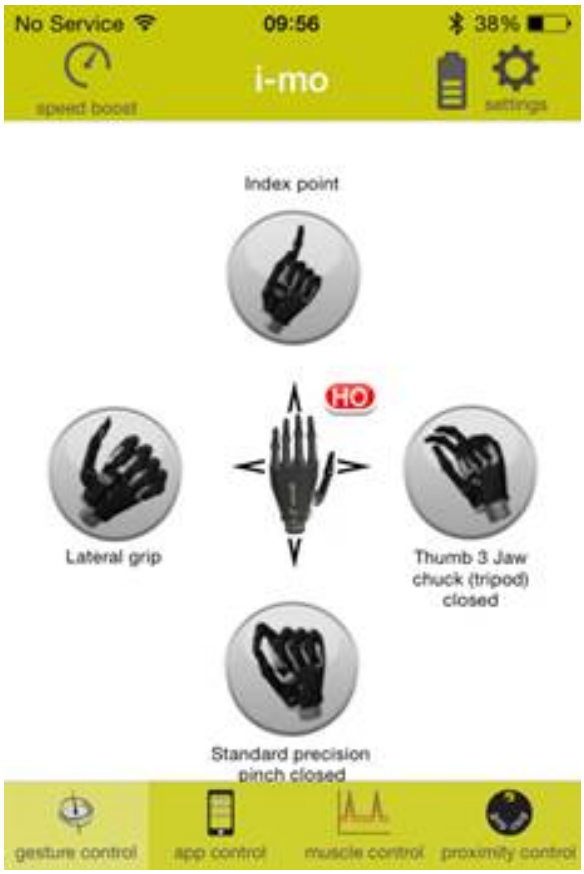

Figure 2.5 Touch Bionics i-mo interface for associating grip patterns with accelerometer based gestures. Image from [34].

It is possible to extend EMG commands issued to modern robotic prosthetic systems by either going through a hierarchical control scheme (e.g., co-contract muscles, move down a "menu" to select a different mode), use a manufacturer provided tablet or smart phone application with the intact hand, or use time based input (e.g., contract muscle for longer than two seconds to switch to a different operation mode). Unfortunately, none of these systems are as seamless in operation as most users would like. On the other hand, multiple input EMG classifier systems discussed in section 2.5.2 have the promise of allowing the user to command multiple degrees of freedom

at the same time.

A promising surgical approach called Targeted Muscle Reinnervation (TMR) [36] [37] [38] (Fig. 2.6) involves surgically connecting the residual nerves from the amputated arm to chest muscles and using an EMG classifier system to enable the control of multiple degrees of freedom in the prosthetic arm. In this scheme, the subject wishing to command a non-existent muscle in the amputated limb instead activates a muscle in the targeted area (in this case, the chest). This method utilizes the body's on muscles as amplifiers for the nerve signal, and has been used to control multiple degree of freedom devices. It is, however, invasive. In addition, not many surgeons are currently qualified to perform the required procedure.

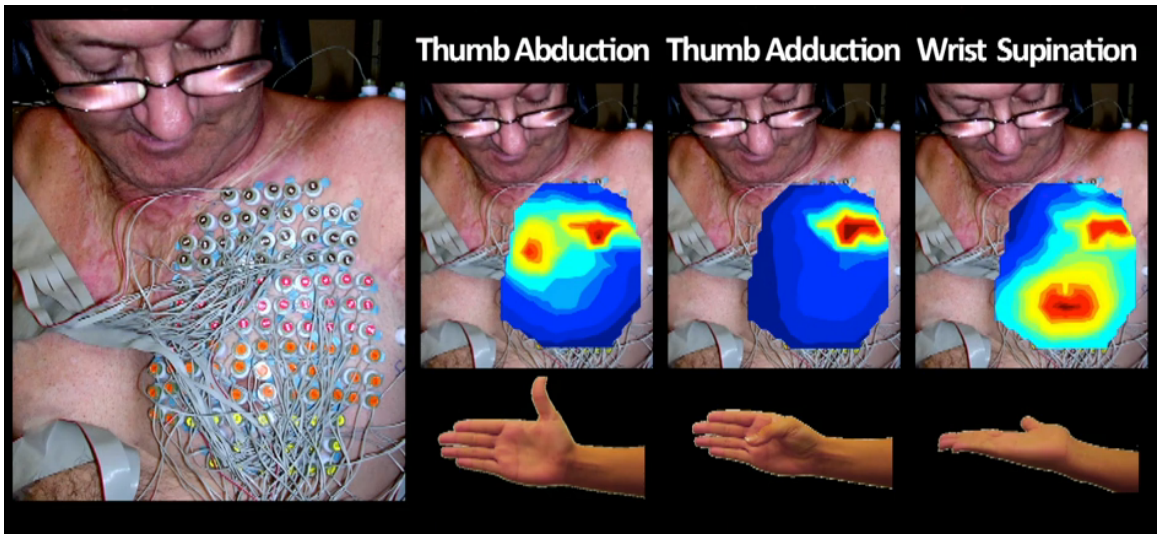

Figure 2.6 EMG Classification from Targeted Muscle Reinnervation (TMR). Image from [39].

Direct peripheral neural interfaces in which the electronic systems are interfaced to residual nerves via neural electrodes, as well as electrocorticography based Brain Computer Interface systems also hold promise for the future, but are currently not deployed due to their invasive nature and instability. A survey of such neural interfaces for commanding prosthetic devices can be found in [40] as well as in [3] [4] [5] [6] and [7]. Such systems will not be considered further in this work.

2.3 Healthy Limb Based Control

This section discusses enabling technologies for the healthy limb based control systems created, as well as the challenges presented in healthy limb based device control.

Double amputations are rare, therefore the majority of upper limb amputees are unilateral (i.e., they have only lost one arm) [41]. For amputees who are unable to generate proper EMG signals, the healthy limb based control paradigm presented here will allow the existing hand to command the prosthetic device worn on the residual limb. The most obvious objection raised to this scheme is that the healthy limb would be occupied commanding an inferior robotic device instead of performing the task. This objection is countered in two ways: 1) A subset of tasks require both hands to be performing mirrored versions of the same action, such as when grabbing a laundry basket, table, or other large object [42] and 2) A majority of the tasks performed by the amputee involve holding an object in the prosthetic hand while performing dexterous manipulations with the intact hand (such as when opening a jar, a padlock, etc.).

2.3.1 Existing Systems

2.3.1.1 Smart Phone Based Prosthetic Control

Several companies, such as TouchBionics [8], provide a smart phone app to control the prosthetic device. This app is intended to familiarize amputees with the capabilities of their device while they learn how to generate proper EMG signals to drive the device, as well as posing the device in pre-specified poses and changing the modes of operations of the device.

2.3.1.2 Rotatable Wrists

In addition, many prosthetic devices have a rotatable wrist, allowing amputees to use their intact hand to rotate the wrist in order to position the robotic hand in the right position to perform their task (such as a 90 degree rotation in order to properly grab a coffee cup). Automated wrist rotators do exist, but require the availability of an extra degree of freedom in the input, or for the user to go through a set of hierarchical values to go into a "wrist control" mode.

2.4 Enabling Technologies for Healthy Limb Based Control

The systems presented in this work are meant to use the healthy limb to easily command the prosthetic device in several modes: by mirroring the intact hand, by putting the device in several pre-specified grasps (such as pinch grasp, lateral grasp, etc.), and to put the device in different modes of operation without having to interact with an on-screen menu system. This section discusses technologies on which the newly developed healthy limb based control systems are based.

2.4.1 State of the Art in Hand and Gesture Tracking Technology

2.4.1.1 Cyberglove and similar kinematic tracking systems

Since the advent of Virtual Reality technology in the 1990s, there have been multi-joint kinematic hand tracking systems such as the $CyberGlove^{TM}$. This device, originally developed as "The Talking Glove" by Kramer et al. [43] as a communication device for the deaf, can have anywhere from 18 to 22 bend sensors embedded inside it to track various joints [44]. This system is frequently combined with the CyberGrasp [45] haptic feedback system (see Fig. 2.7) for closed loop teleoperation of robotic devices.

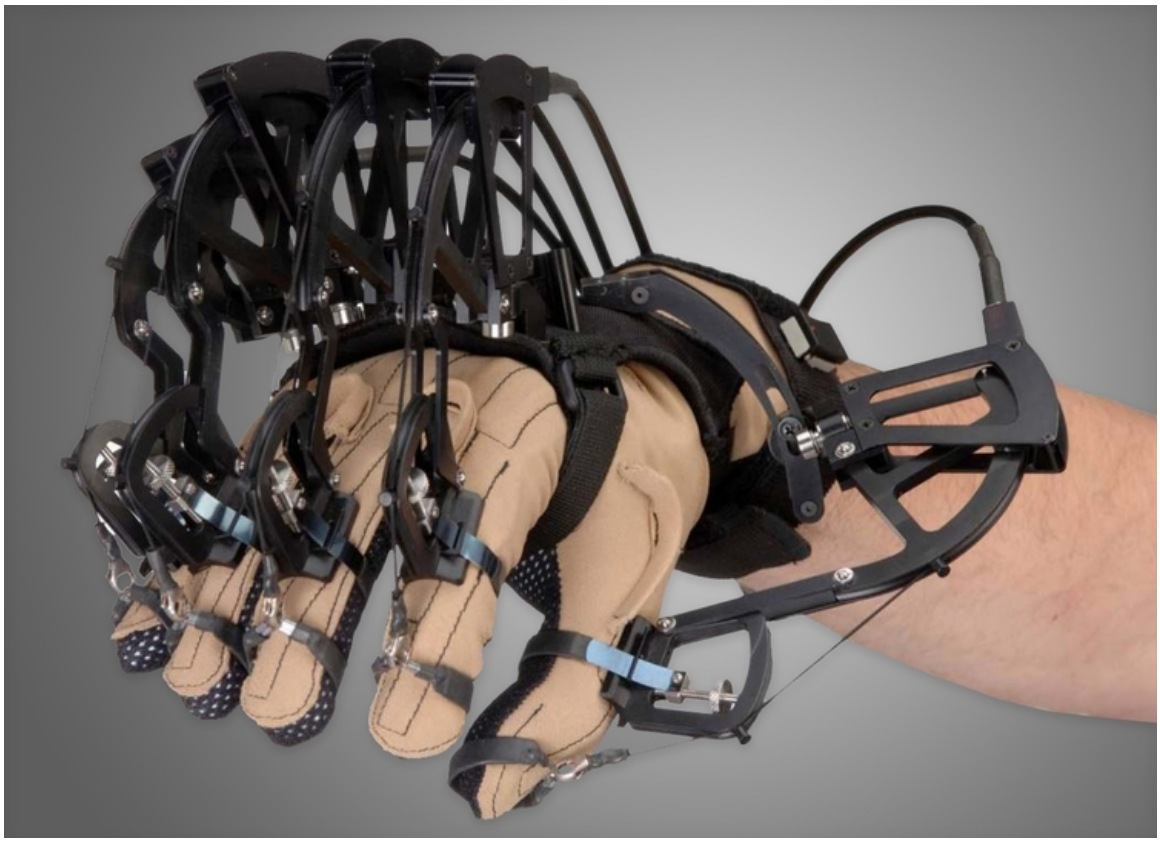

Figure 2.7 CyberGlove hand tracking system with CyberGrasp exoskeleton. Image reproduced from [45].

2.4.1.2 Pinch Glove Based Systems

Another system originally designed for the virtual reality market is the pinch glove system [46] in which the user inputs discrete events into the computer by using the fingers to touch various parts of the glove. The current systems on the market are typically created using conductive fabric and require that the user wear a glove covering the entire hand.

2.4.1.3 Computer Vision Based Systems

In recent years, computer vision based systems such as the Leap Motion System and the Microsoft Kinect One system [47] (which is based on a Time of Flight camera to determine depth) have been used for gesture recognition [48]. While computer vision based systems have the advantage of not requiring the user to wear anything, both the Leap Motion system and the Kinect system require that the users be present in a pre-determined environment and are not designed as wearable gesture tracking devices. On the other hand, Kim et al. [49] (Fig. 2.8) describe a wrist-worn camera based system which can track and interpret gestures to control various devices such as MP3 players. Among the others mentioned so far, the system is unique in that it promises to enable wearable interaction.

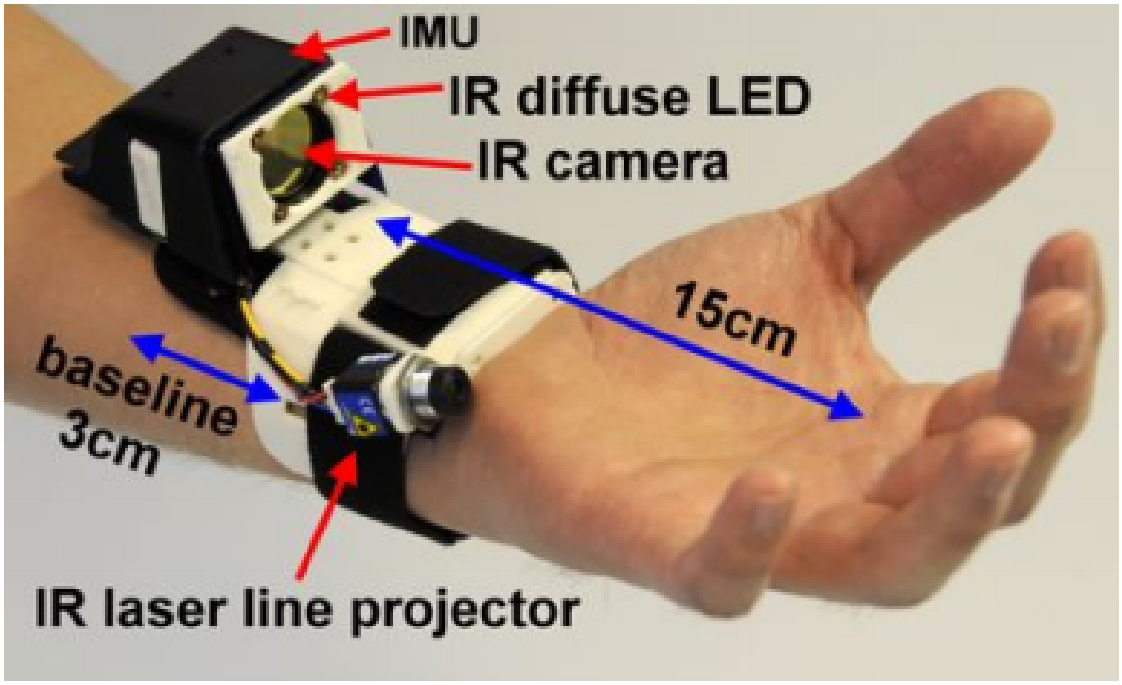

Figure 2.8 Wrist mounted camera system for gesture tracking from Kim et al. [49]. Image reproduced from [49].

The wrist worn camera system is also prone to occlusions if the hand is oriented in a way which is not in view of the camera.

In recent years, the emphasis has shifted to wearable systems which are meant for portable device control. Two notable examples are the Myo^{TM} armband and the *gest* system.

2.4.1.4 Myo Armband

The Myo armband [50] [51] 2.9 is a commercial EMG and accelerometer based device intended to be a substitute for mouse and touch screen based interfaces. The Myo is an interesting system in that it utilizes a combination of accelerometer and EMG technology for gesture tracking outside the context of prosthetic control. This

device has been used to control games, PowerPoint presentations, as well as mobile devices such as cell phones and portable music players. In order to utilize the system properly, the user has to learn the gestures which are understood by the system. One example is controlling the volume of a music player by clenching the fist and rotating the arm clockwise, as if turning a volume dial.

Figure 2.9 The Myo armband is a gesture input device utilizing eight EMG sensors as well as accelerometers.

Recent work [52] has investigated the use of the Myo armband to control a prosthetic device using gestures which are custom made for the Myo.

2.4.1.5 The *gest* System

The *gest* glove [53] relies on small tethered rings which are worn on each finger as seen in Fig. 2.10. Each ring is instrumented with accelerometers. This allows the system, along with some machine learning algorithms, to recognize gestures.

Figure 2.10 The *gest* glove.

This system has the advantage of leaving the fingers free of haptic encumbrance, but still requires each tracker to be tethered to the main device.

2.4.2 Gesture Recognition Systems Developed

For this work, three custom gesture recognition systems have been developed, which will be detailed in Chapter 3 and Chapter 4. The Mirroring Glove is a kinematic tracking system relying on piezoresistive flexion sensors to track up to 6 degrees of freedom. While more can be achieved, this system is sufficient to drive the 6 degree of freedom commercial prosthetic device used in this work.

The Gesture Glove is a unique system similar in principle to the Pinch Glove [46] but leaves the fingers free. It accomplishes this by taking advantage of conductive pads embroidered in the palm of the hand, very small amount of current being carried by the skin through the bare fingers, and signal processing.

The Fingernail Device is a novel fingernail worn gesture input system created in the course of performing this work. This system is unique in that it can be made into a low cost, completely untethered system which can be worn as fake fingernails on each finger, with a slightly larger device worn on the thumbnail in order to recognize gestures free of haptic encumbrance. This is somewhat similar to the system proposed by Mascaro et al. [54] which relies on the color change on the fingernail bed, but has the advantage of supporting an easily untetherable implementation and not requiring any contact force between two fingers touching each other. This system relies on the fact that light can be transmitted from one fingernail with an LED pointed to it to the thumbnail only when contact is achieved between the two digits.

2.4.2.1 Light Transmission Through Tissue and the Near-Infrared window

The Fingernail Device, described in Chapter 4, relies on the transmission and detection of light through human tissue. Even though most biological tissue appears

to be opaque in everyday life, certain wavelengths of light are transmitted through tissue (as can be observed if a bright flashlight is held up against one side of the hand). This range of red and infra-red frequencies is known as the "near infrared window" in biological tissue. Light between the wavelengths of 670 nm and 910 nm, and above the wavelength of 1050 nm [55] is transmitted through tissue as illustrated in 2.11.

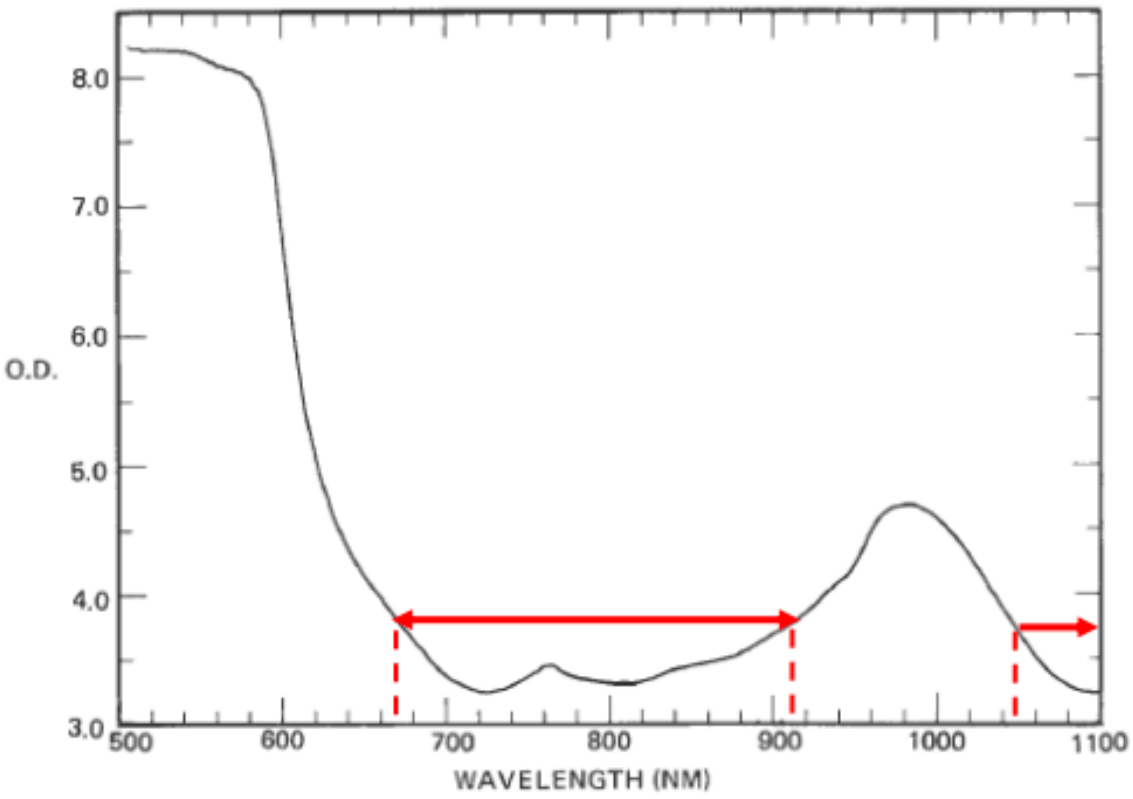

Figure 2.11 Light Transmission through the hand. Reproduced from [55].

The transmission and scattering of light through biological tissue is a very complex matter whose mathematical treatment is out of the scope of this work. A

good treatment of the subject can be found in [56]. Further analysis of the way a light pulse interacts with tissue can also be found in [57].

The attenuation of light in a homogeneous medium is exponential in nature and can be approximated through the Beer-Lambert law [58], reproduced in Equation 2.1 below. Even though living tissue is not homogeneous and the actual attenuation and scattering behavior is slightly different, Equation 2.1 illustrates the fundamental problem of exponential light loss encountered in the design and construction of the Fingernail Device.

$$I = I_o e^{-\mu x} \tag{2.1}$$

In Equation 2.1, I_o represents the incident light on the system, μ represents the extinction coefficient, and x the distance light travels through the system. Cheong et al. [59] give a comprehensive review of various extinction coefficients of human tissue. For the design of the Fingernail Device system, the LED wavelength, intensity, and detector amplification have to be chosen so that light traveling through a finger and a thumb is still detectable through the thumbnail. The Fingernail Device is discussed in Chapter 4.

2.5 Electromyography

2.5.1 Electromyography Signal Acquisition and Processing

Electromyography (EMG) is the recording of electrical action potentials from muscles [60]. Typically, surface EMG (sEMG) signals are obtained and amplified from electrodes placed on the skin over the muscles. It is also possible to obtain more refined signals by using needle electrodes which are inserted inside the muscle. While EMG has many diagnostic uses in medicine, it has also found an essential niche in

prosthetic device control. Most commercially available devices are controlled via two EMG electrodes placed on the residual muscles of the user, through which the user is trained to issue commands.

2.5.2 EMG Based Classification Algorithms

Commercial prosthetic devices available at the time of writing typically exploit only a subset of the available information for robotic prosthetic device control, allowing the users to issue at most two commands, which can be used to drive menus of options through hierarchical control [61] which is both difficult to learn and time consuming to execute. A major area of research, therefore, has been the use of multiple sEMG sensors on the residual limb in order to enable multi-DOF control.

The data from multiple sEMG sensors are typically used along with machine learning systems which can classify the input into discrete actions (such as pronating or supinating the hand, or moving an individual digit). These classification systems generally can't be explicitly programmed, but are able to decide what types of data belong together (unsupervised learning systems), or are trained on existing data sets in which the "ground truth" is known (supervised learning systems). This work focuses on supervised learning systems in which the ground truth is obtained from tracking the intact hand during training while the subject performs mirrored bilateral movements. Once a classifier is properly trained, it can map muscle movements to actions performed by the prosthetic limb in real time.

Surveys of machine learning systems for sEMG classification can be found in Karlik et al. [62]. Several implementations include those by Soares et al. [63] and Nair et al., [64]. A well known system has been implemented by Engelhart and Hud-

gins [65] which extracts four degrees of freedom from multiple EMG sensors. Others algorithms include those referenced in [66] [67] [68] [69] [70].

2.5.3 The Need for a Better Classifier Training Scheme

Despite the promise of sEMG classification, the systems have not yet been widely deployed. Part of this stems from the fact that most classification experiments have been run in the lab, with the patient's residual limb resting on a table. Unfortunately, the systems trained under such ideal conditions do not perform well in daily use. Scheme et al. [71] report on the effect of arm position when training data are being acquired, and Hargrove et al. [72] report on the degradation of signal due to non-ideal electrode placement.

Before sEMG classification systems can be moved out of the lab and into clinical use in any meaningful way, a classifier training method needs to exist which can be re-trained by the user as needed, without the need for a laboratory environment. The ability to train the system under various arm poses is also important as established by [71]. On a related note, any deployed system will ultimately need to take arm pose and position into account both during training and use.

2.5.4 Existing Systems of Relevance

2.5.5 Use of Bilateral Movements for Training

Several groups (e.g., Nielsen et al. [73], Jiang et al. [74], Mucelli et al. [75]) have demonstrated using mirrored, bi-lateral movements to associate forces and movements measured in the sound limb with EMG signals in the amputated limb.

2.5.6 Optimal Electrode Placement

Existing work by Maier et al. [76] describes optimal sensor placement for finger movement classification, which has been reproduced in Fig. 2.12. As illustrated, 1) Flexor pollicis longus (thumb flexion) 2) Flexor digitorum superficialis (index finger flexion) 3) Flexor carpi ulnaris (pinkie flexion) 4 to 7) Flexor carpi radialis and palmaris longus (middle and ring finger flexion) 8) Extensor pollicis longus (thumb extension) 9) Extensor indicis (thumb extension) 10) Extensor carpi ulnaris (middle, ring, pinkie extension).

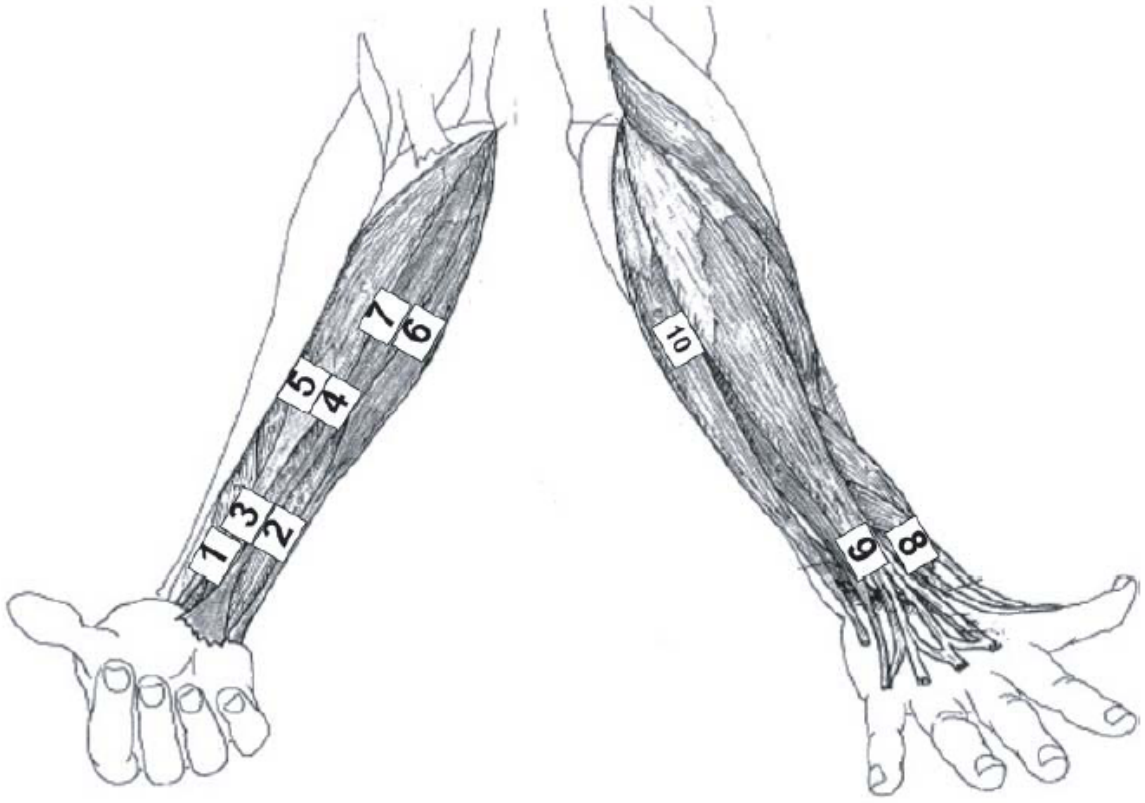

Figure 2.12 Sensor Placement from Maier et al. [76]

2.5.7 Dimensionality Reduction

Optimal electrode placement, while possible in the lab, presents a unique challenge to an amputee who must put on and wear a prosthetic arm on the amputated limb using a single hand. Furthermore, it is established that deviation from optimal electrode placement will degrade the performance of a classifier system [72]. An alternate approach to accurate electrode placement is to use a multitude of electrodes and then only utilize the ones which provide relevant data. Two approaches are possible to accomplish this: 1) Reduce the dimensionality of the data via the use of a Support Vector Machine or a similar technology 2) Use an Artificial Neural Network on the entire dataset and allow the network training algorithm to reduce the significance of the unused inputs.

Several groups have thus made use of multiple electrode arrays. For example Sueaseenak et al. [9] have used a 16 electrode array to classify 4 different movements (their array setup is reproduced in Fig. 2.13).

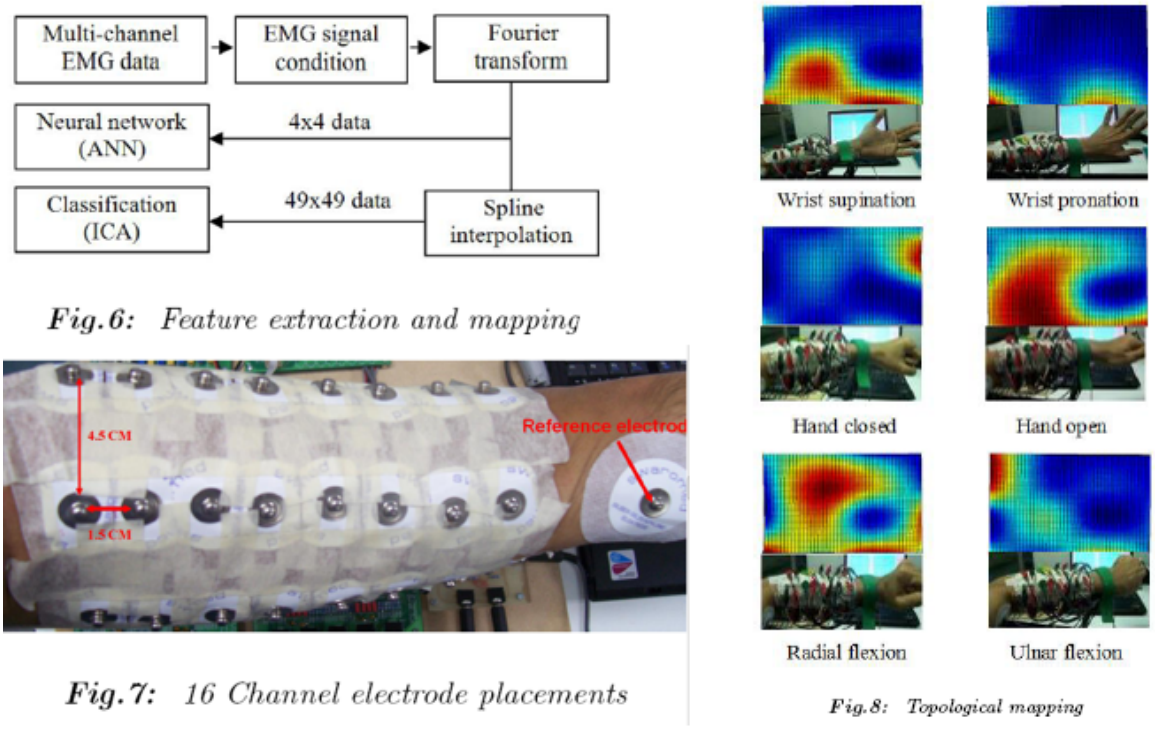

Figure 2.13 Multiple EMG sensor array from Sueaseenak et al. [9].

2.6 Force Myography

An alternative to EMG based control is Force Myography (FMG) [77, 78, 79] (alternatively referred to as Residual Kinetic Imaging) in which the volume changes in muscles and tendons in the forearm or the residual limb are recorded and correlated with desired movements.

Force myography is simpler in some ways than EMG, since the data do not need to be filtered as much and the signal is not affected by factors such as skin conductivity. There is also the advantage that FMG may be able to access tendon movements which do not necessarily have a readily accessible EMG counterpart.

In this work, EMG and FMG are used synergistically in order to improve classification accuracy.

CHAPTER 3

HEALTHY LIMB AND GLOVE BASED INTERFACES

3.1 Introduction

Commercially available robotic prosthetic limbs with multiple degrees of freedom (DOFs) closely resemble the human hand [80] [36]. Most interfaces rely on EMG pattern recognition or hierarchical control [61] allowing only single actions at a time. Control difficulties [81] and the inability to generate the required EMG signals [40] lead to a 20% incidence of device abandonment [82]. Some manufacturers supply a smart phone app for intact hand based control while users acclimate to EMG [8]. We present a control methodology for unilateral amputees who cannot use EMG signals or who desire to exploit a higher number of DOFs.

3.1.1 System Architecture

Two control systems using wearable electronics on the intact hand were created and tested. Data from both control systems are read by an $ArduinoUno^{TM}$ microcontroller board and relayed to a $RaspberryPi^{TM}$ mounted on the prosthetic device as illustrated in Fig. 3.1.

The system relies on a modular architecture in which a standard set of commands are issued using the RS232 protocol transmitted by a USB cable to the embedded computer, which then relays the commands to the prosthetic device via the CANBus protocol. The Arduino microcontroller on the glove interprets the input and creates UNIX shell commands which are then communicated to the embedded

computer (Raspberry Pi) via USB. These messages are in turn executed, causing the command to be sent to the prosthetic device over the CAN interface.

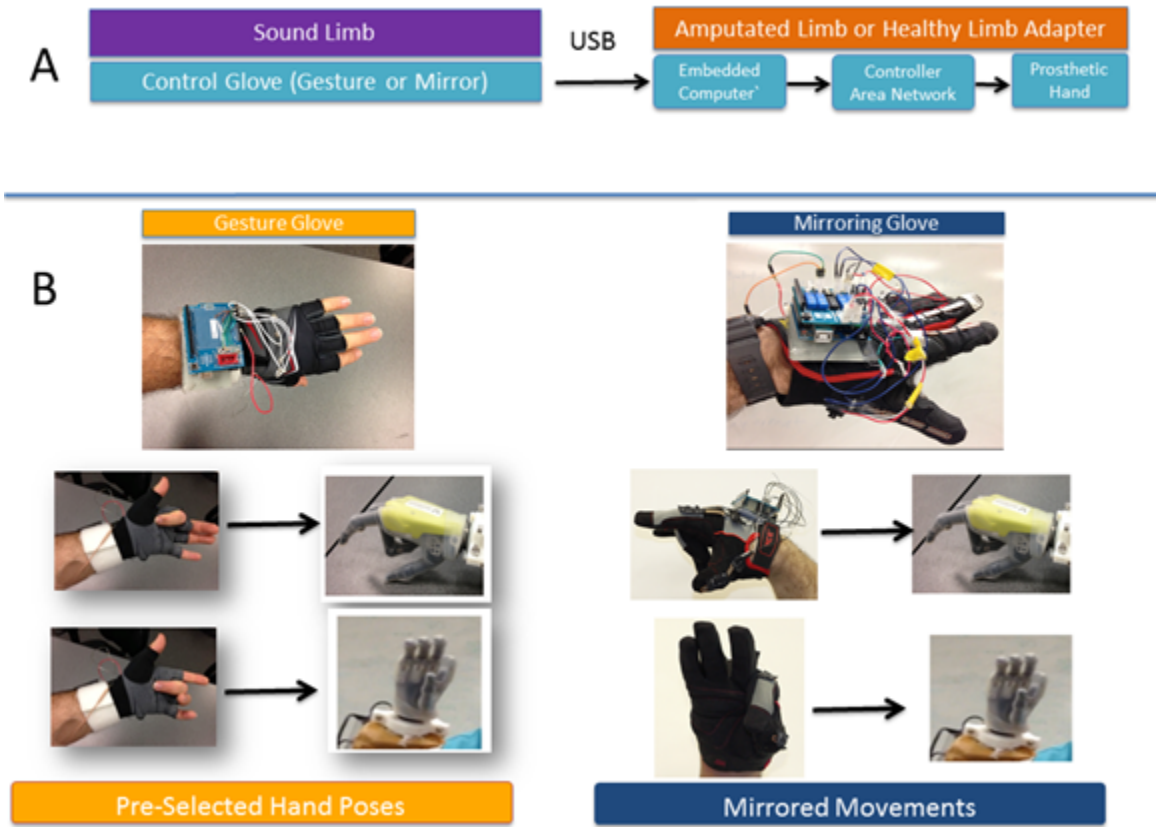

Figure 3.1 System Architecture. A: Glove Based Control Architecture. Either glove is worn on the sound hand. B: Left: The Gesture Glove and two gestures, mapped to two different poses. B: Right: The Mirroring Glove and two gestures being mirrored on the prosthetic limb.

3.1.2 Healthy Limb Adapter

In testing on healthy volunteers, the prosthetic device (robo-limb, Touch Bionics; Livingston, United Kingdom) was mounted on a custom-made healthy limb

adapter (Fig. 3.2).

In order to allow non-impaired volunteers to wear and test this system, a healthy limb adapter was created incorporating an Otto Bock Quick Connect (Otto Bock, Germany) ring attached via six standard nuts and bolts to a custom 3D printed bracket attached to a plastic shell. The shell was formed out of heat moldable thermoplastic (Worbla, USA) with a heat gun by using a plaster of Paris replica of the subjects arm as a template. Power and data wires were routed to the outside of the healthy limb adapter through a drilled hole. Circuit boards necessary to terminate the controller area network (CAN) cable and provide power, batteries, and a Kvaser Leaf (Kvaser, USA) USB to CAN adapter were affixed to an armband using Velcro. A Raspberry Pi embedded computer was also mounted to interpret the signals from the Arduino Microcontroller and generate the CAN messages driving the prosthetic device. A 2 meter long USB cable connects the tracking glove to the Raspberry Pi device on the healthy limb adapter.

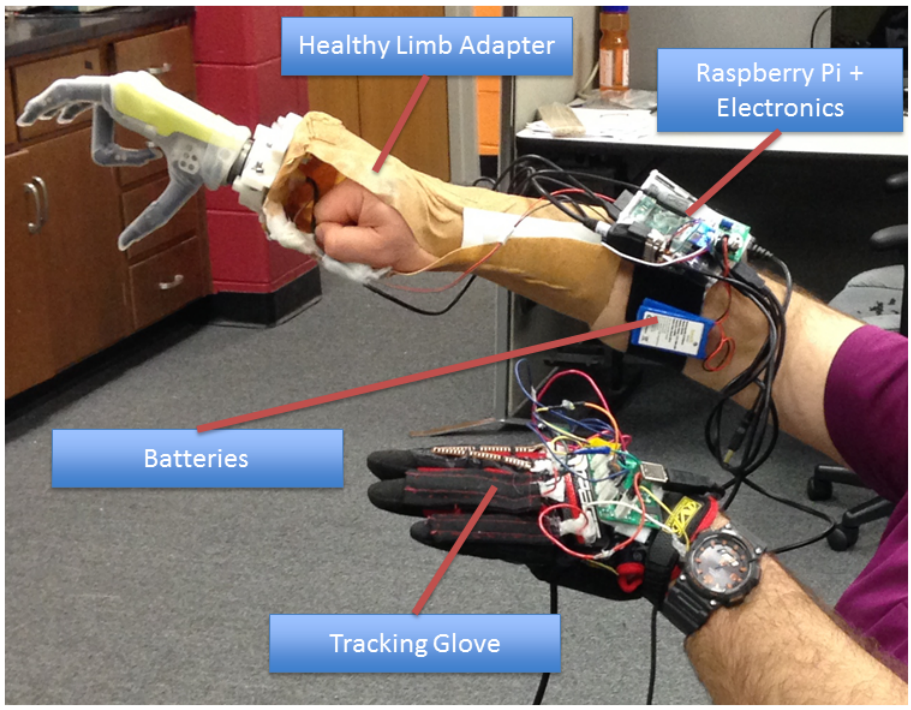

Figure 3.2 Healthy Limb Adapter.

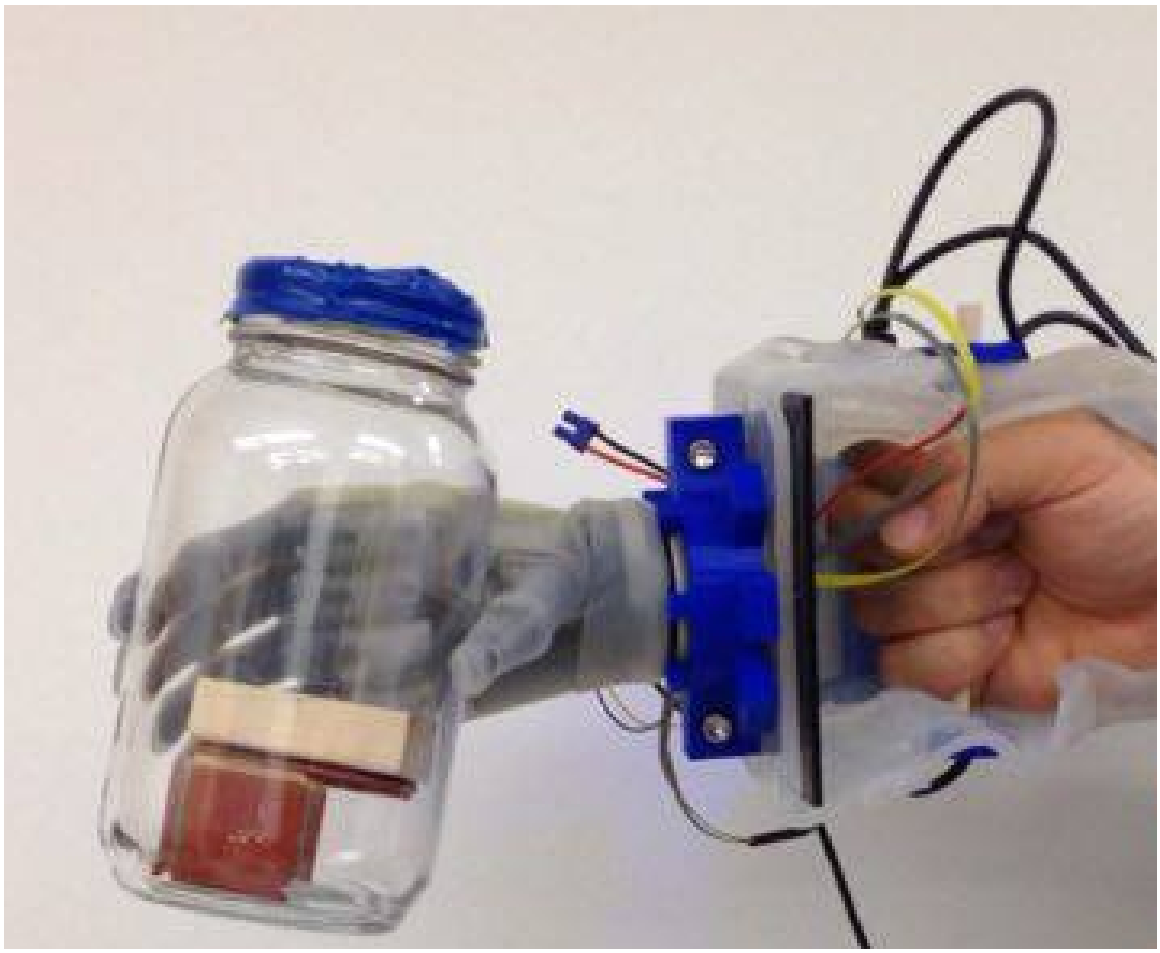

Figure 3.3 Updated version of the healthy limb adapter featuring a handhold and a softer OrthoFlex shell.

3.2 Mirroring Glove

The Mirroring Glove is a 6 Degree of Freedom (DOF) kinematic hand tracking device used to teleoperate the Touch Bionics RoboLimb device inspired from earlier "dataglove" systems [83] which have been used for decades to interact with virtual environments. The main impetus behind the Mirroring Glove concept is the fact that the best input device for a hand-shaped device is the hand itself, and that certain activities of daily living involve "mirrored" movements of both hands. These activ-

ities include, but are not limited to carrying laundry baskets and picking up large awkward objects as seen in Fig. 3.4.

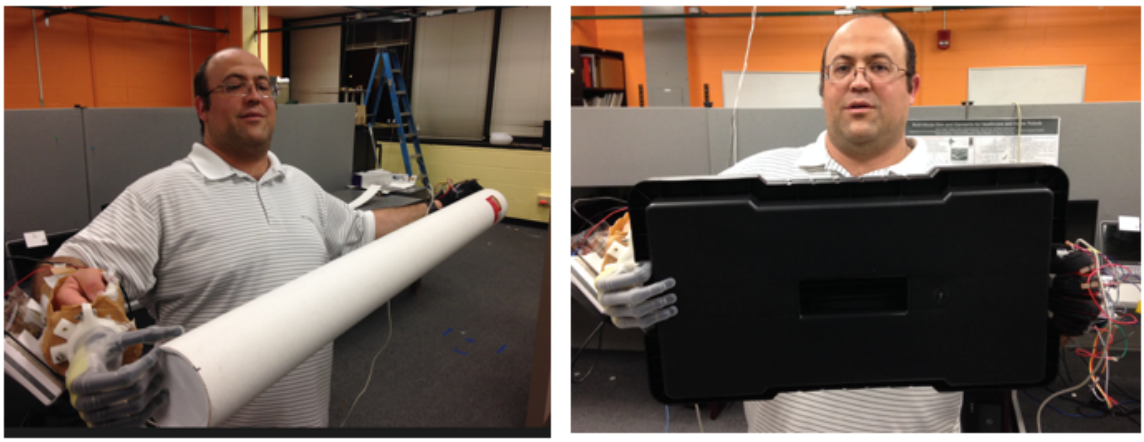

Figure 3.4 Some typical Mirroring Glove uses.

3.2.1 Mirroring Glove Construction

The Mirroring Glove itself is comprised of flexion sensors, an amplifier board to read the sensors, and a glove onto which these sensors are sewn or otherwise attached. An early example of the device can be seen in Fig. 3.5. The Arduino microcontroller on the glove reads the flexion signals. The lock button allows the user to pose the prosthetic in a desired position and lock it in place, making non-mirrored tasks also possible with the system.

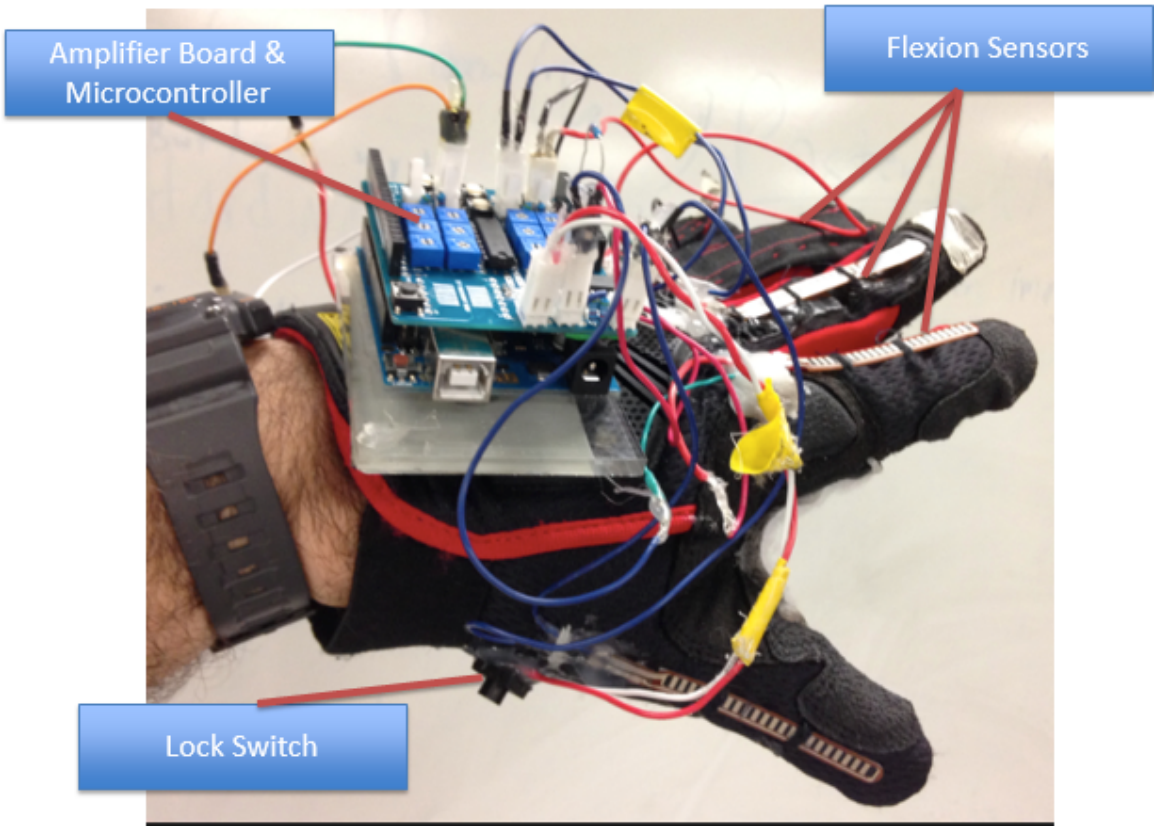

Figure 3.5 Mirroring Glove with lock button.

We have built a tracking glove using six SparkFun 2.2 flexion sensors (SparkFun Inc.), a glove, an Arduino [84] Uno microcontroller, and Velcro to hold the Arduino device on the glove. The flexion sensors were attached to the glove through the use of custom-sewn sleeves which reduce the stress on the sensor attachment point by allowing the sensor to slide during movement. The flexion sensors were soldered to flexible multi-strand wires, and the contact areas were thermoplastic encased. The endss of the sensors were also molded into sewable buttons by a process involving laser-cut acrylic molds described in Fig. 3.6.

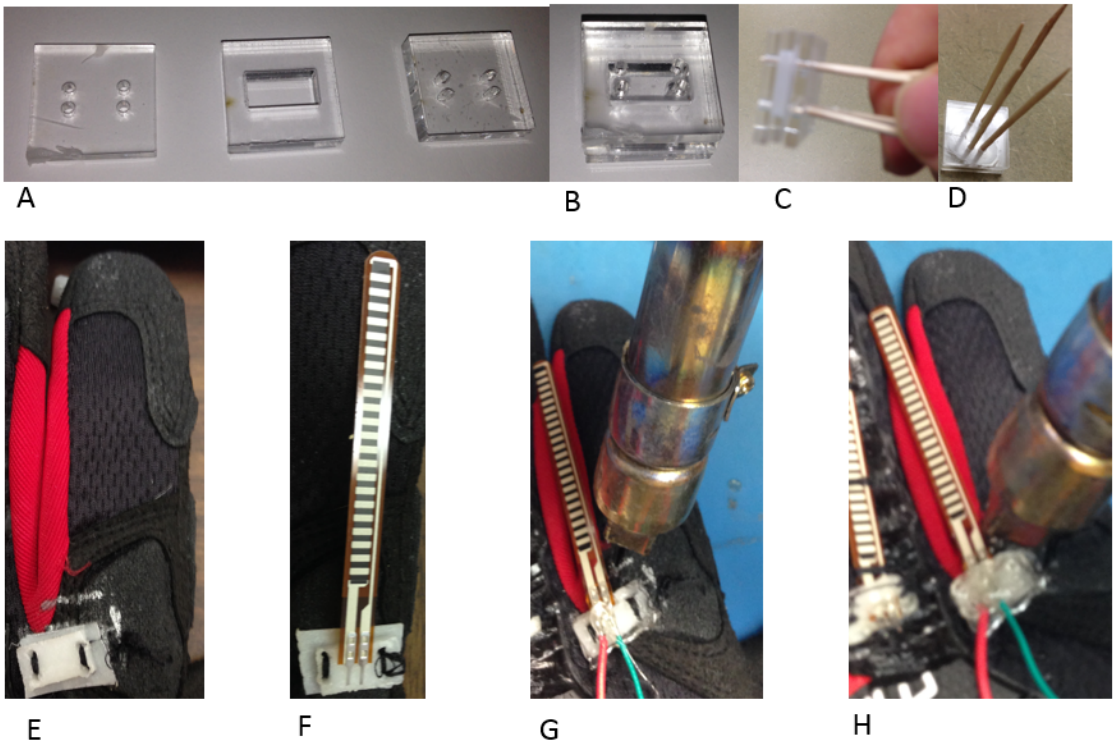

Figure 3.6 Steps involved in making a sewable sensor holder.

As illustrated in Fig. 3.6 A-D: Three layer acrylic mold created with CNC laser cutter used to mold a button with $InstaMorph^{TM}$. E: Molded button sewn onto glove. F: Flexion sensor lined up with finger. G,H: Hot air gun used to immobilize sensor after contacts have been soldered.

3.2.2 Mirroring Glove Software

The software configuration of the system relies on three components: 1) The Arduino microcontroller, 2) the standard UNIX BASH shell, and 3) code running on the Raspberry Pi device (handControl) which can open and close individual digits via the command line. The system operates as follows: first, code running on the

microcontroller board reads the flexion values for each digit and maps them to a value between 0 (open) and 100 (completely closed). Next, the microcontroller software composes the necessary UNIX command to set the position of each digit and sends it over the USB cable to the Raspberry Pi. After that, the Raspberry Pi pipes the command received from the serial port to the BASH command interpreter, invoking the handControl software with the necessary arguments. Finally, the handControl software issues the command to stop any movement that might be happening on the prosthetic device, followed by the command to instruct each digit to assume the desired position.

3.2.3 Mirroring Glove Testing

Testing the Mirroring Glove included characterization of the sensors in order to make sure that they can drive the device in a reproducible manner. Fig. 3.7 shows the output of flexion sensors on repeated hand openings and closings.

3.3 Gesture Glove

The Gesture Glove developed for this system is an alternative as well as a complement to healthy limb based prosthetic device control using the Mirroring Glove. Instead of commanding the prosthetic device via teleoperation, the Gesture Glove allows the mapping of certain common grap patterns to gestures performed on the healthy hand by utilizing the Gesture Glove (as seen in Fig. 3.8). Note that the gesture performed by touching the finger to the palm of the hand does not necessarily correspond with the action performed by the prosthetic hand.

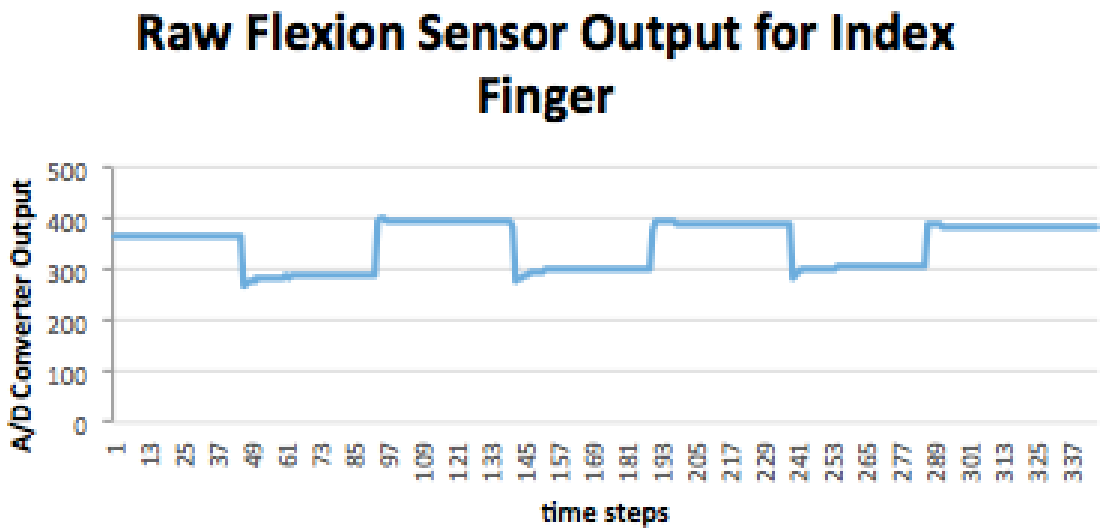

Figure 3.7 Raw data from flexion sensors on one finger on repeated opening and closing of the hand.

3.3.1 Gesture Glove Construction

A conductive thread and glove based system (The Gesture Glove) was also constructed and investigated. This system leaves the fingers free for the performance of non system-related tasks (Fig. 3.8). The Gesture Glove system was developed to be less cumbersome and is suitable for being worn for extended periods of time. The Gesture Glove was constructed from a fingerless weightlifting glove by embroidering conductive thread touch pads in the palm and connecting them to snap connectors sewn on the dorsal side of the hand. The conductive thread lines going from the embroidered pads to the snap connectors were insulated with fabric affixed to the inside of the glove. Another contact pad was embroidered on the inside of the glove so that it made contact with the skin and was connected to the +5V terminal of the Arduino microcontroller board via another snap connector. The user touches one of the conductive pads with a finger and completes the circuit causing a signal to

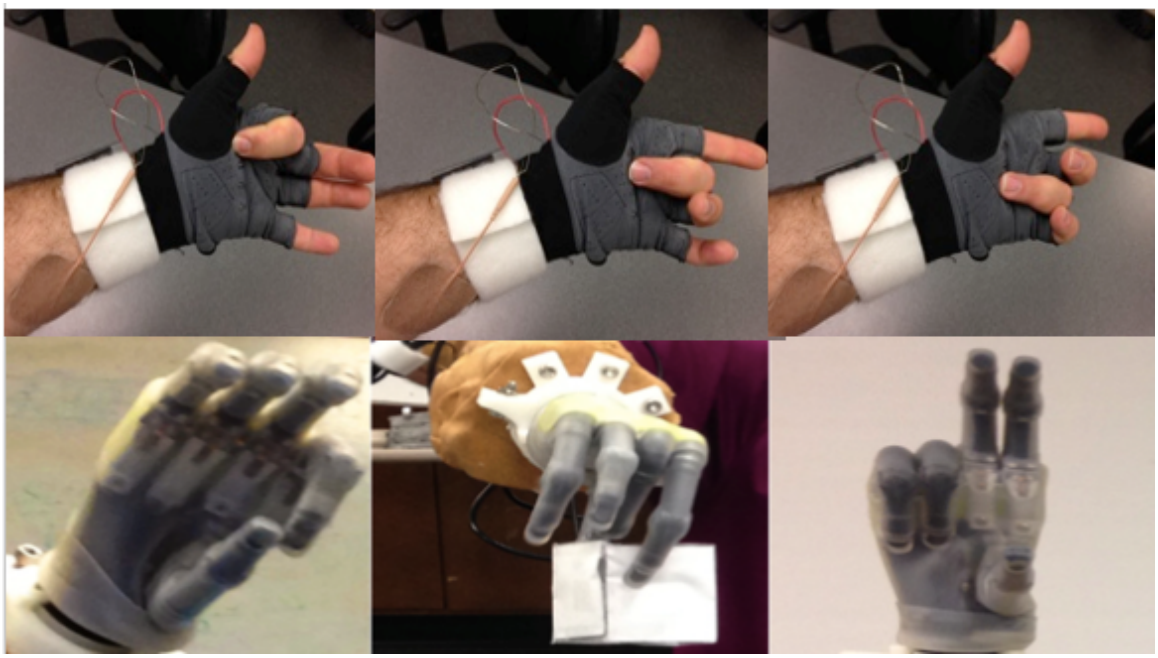

Figure 3.8 Gesture Glove with some possible actions.

be acquired by the A/D converter. The purpose of the system is to identify touch gestures from each fingertip while not registering any accidental gestures from objects that the user might be manipulating. The skin contact with the +5V terminal was further enhanced by connecting the terminal to a gel electrode meant for use with a Transcutaneous Electrical Neural Stimulator (TENS) system (Infiniti ELT 5050T). The TENS electrode is not strictly necessary, but significantly improves the signal to noise ratio, essentially eliminating misclassified gestures. The output from the conductive pads was sampled with the built in 10 bit A/D converter of the Atmega 328 Microcontroller on the Arduino board.

3.3.2 Gesture Glove Operation

In this setup, the hand is treated as a 1 to 2 M resistor, and touching the conductive pad with the bare fingertip completes the circuit, causing the A/D converter to see an increase in voltage. This is depicted in Fig. 3.8.

The data from the conductive touch pad was very noisy, so a touch classification algorithm was developed in order to classify touch events. For this, an algorithm using moving average and moving standard deviation was developed. The moving standard deviation is a method that is useful in various applications where the signal is noisy or the baseline is not stable. Its use and properties have been discussed for various biomedical applications [85, 86, 87]. In this study, an Arduino based statistics library [88] was utilized for part of the code. The touch classification algorithm uses a running average filter and a running standard deviation filter (window size n=50), which is applied to the raw data in real time using only past values. The output value at step i, Oi, is determined to be 100 if a touch event is detected and 0 otherwise. The touch classification algorithm is defined in Equation 3.1 through Equation 3.5 below:

Let the output of the A/D converter be the vector $\mathbf{X}$ of size N

$$\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^{N} \tag{3.1}$$

Let $\overline{x}_{i,n}$ be the moving average at sample i with window n

$$\overline{x}_{i,n} = \frac{1}{n} \sum_{j=i-n}^{i} x_j \tag{3.2}$$

Let $Std_{i,n}$ be the moving standard deviation at sample i with window n

$$Std_{i,n} = \sqrt{\frac{1}{n}\sum_{j=i-n}^{i}(x_i - \overline{x}_{i,n})^2} \tag{3.3}$$

Let A_i be the activation statistic representing the distance of the current value from the moving average in standard deviations

$$Ai = \frac{(x_i - \overline{x}_{i,n})}{std_{i,n}} \tag{3.4}$$

Let O_i be the output from the touch classification algorithm and T_{std} be the threshold for standard deviation (experimentally determined or adjusted with a potentiometer)

$$Oi = \begin{cases} 100 & A_i \geq T_{std} \\ 0 & A_i < Tstd \end{cases} \tag{3.5}$$

After the data were acquired from several trial runs, an activation statistic threshold value, Tstd, of 2.5 standard deviations was experimentally determined to be appropriate for distinguishing between a touch event for a given conductive pad and random noise. In this system, the condition of Oi = 100 in Equation 3.5 holds true for only a few samples after a touch event. After the event, the moving standard deviation changes and causes the value of Oi to be below Tstd again. This has the advantage of eliminating the need to de-bounce the touch event. While it would be possible to classify release events as well as touch events using this method, only touch events are used in the Gesture Glove system. This algorithm has the advantage of being robust against changes in the absolute magnitude of noise and the values read from the contact pads, which change based on factors such as ambient moisture or changes in skin conductivity. The application of the algorithm on a representative dataset (in which four consecutive touches from index, middle, ring, and little fingers were acquired) is illustrated in Fig. 3.9, Fig. 3.10, Fig. 3.11, and Fig. 3.12.

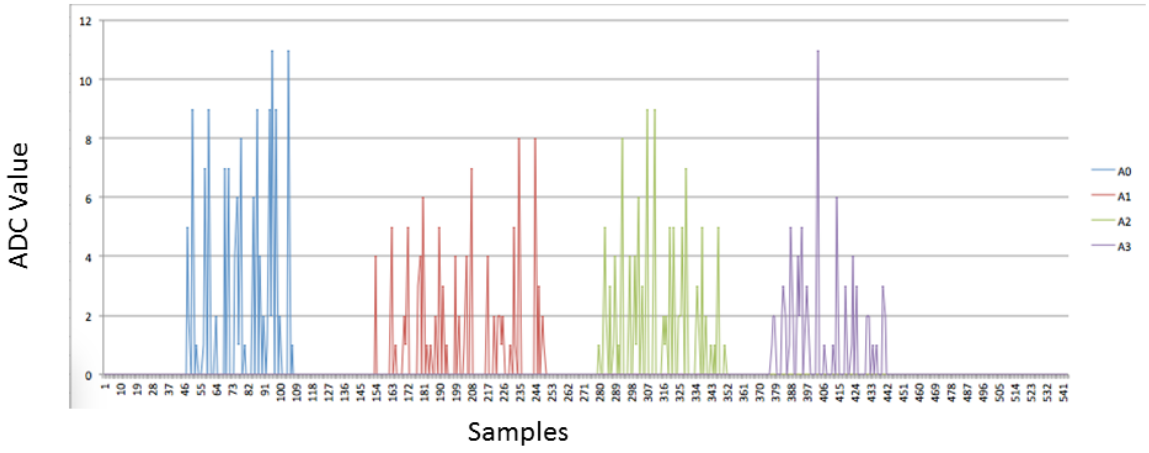

Figure 3.9 Raw data from conductive thread pads. The time series shows consecutive touch events for the index, middle, ring, and little fingers.

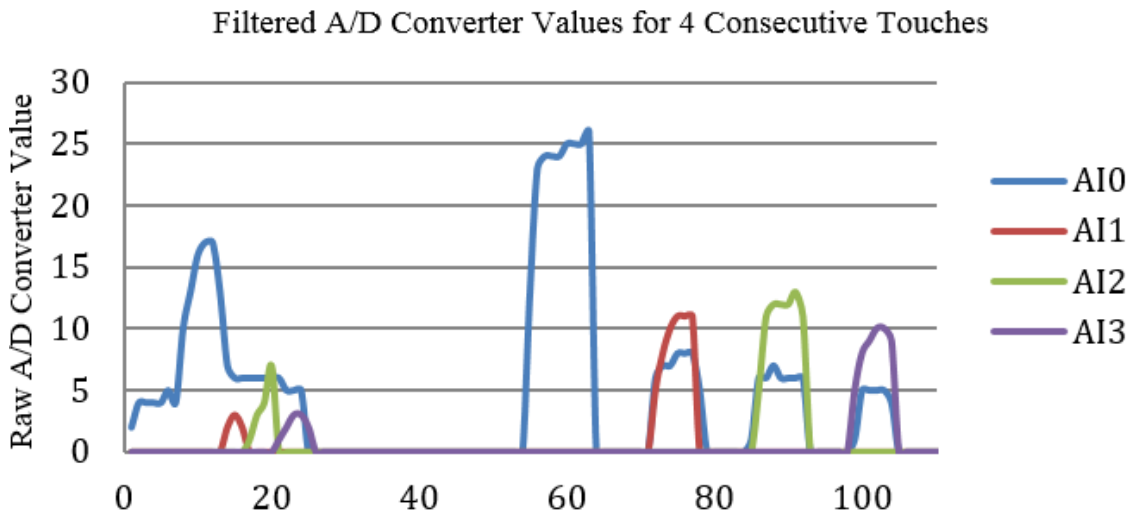

Figure 3.10 Values from A/D converter filtered with a moving average filter with a window size of n = 50, showing four consecutive touch events (index, middle, ring, and little finger).

The Gesture Glove has the advantage of leaving the fingers free and allowing the user to touch items with the palm of the gesture glove without accidentally activating the system. It is, however, affected by sweat and by touching metal objects, which may necessitate more sophisticated signal processing approaches than simply detecting a touch event. The effects of sweat can be mitigated by wearing another lightweight and non-conductive fingerless glove underneath the Gesture Glove.

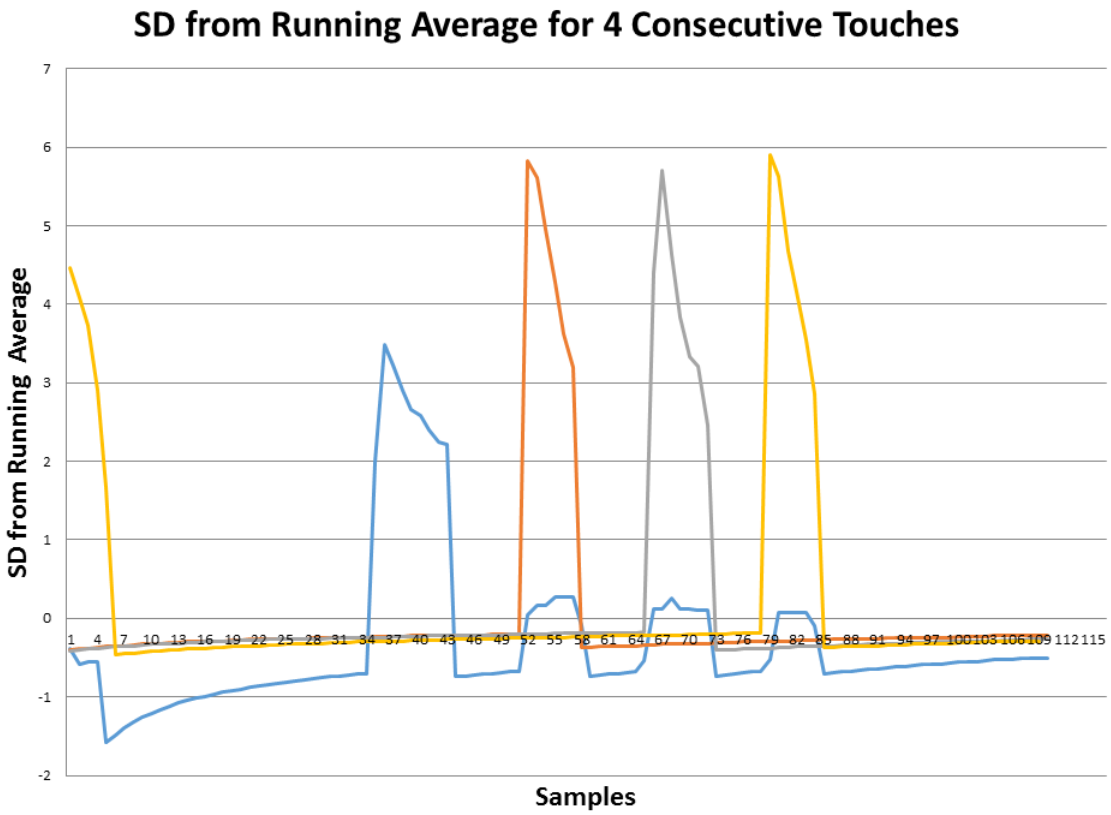

Figure 3.11 The touch events from above displayed as standard deviations from the running average at each point. Note that each touch value goes above 2.5 SD from mean in this case.

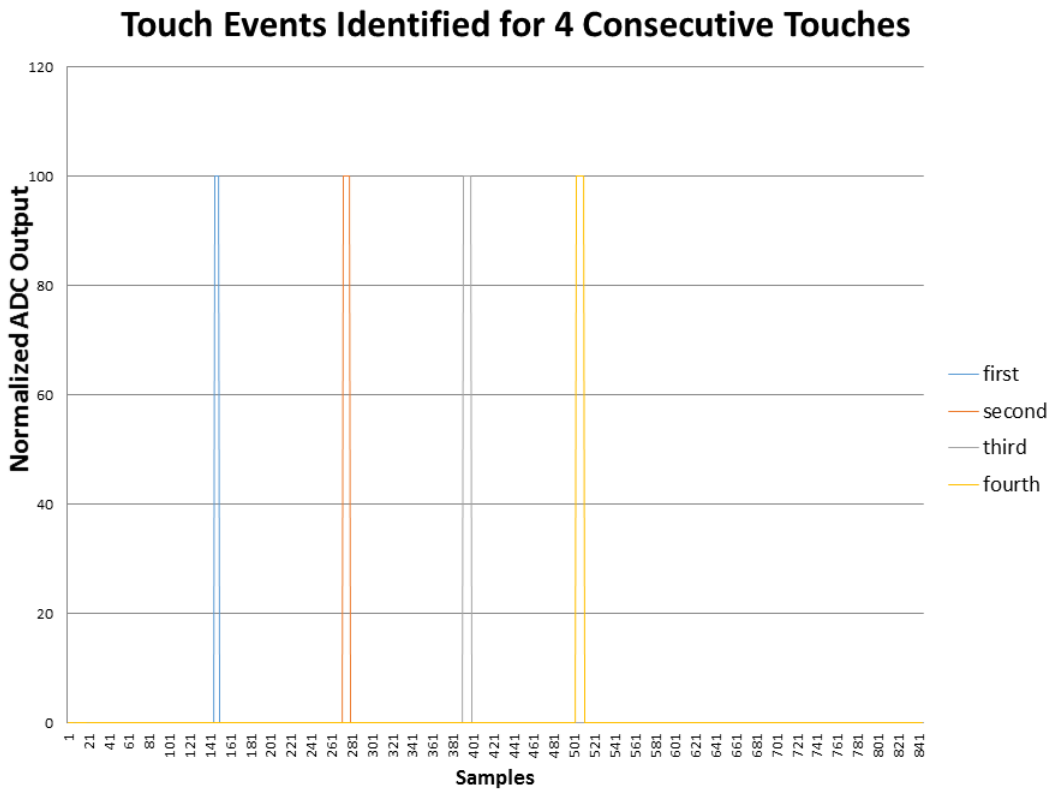

Figure 3.12 Touch events, O, identified after the application of threshold Tstd. The system outputs a value of Oi = 100 (arbitrarily chosen) if the Activation Statistic, Ai, is greater than the standard deviation threshold Tstd. (Note: this figure was obtained from a different run of the experiment than the previous figures).

3.4 Experiments Performed

Four manipulation tasks simulating activities of daily living were utilized: Modified Box and Blocks Test, Blanket Folding Test, Bottle Carrying Test, and Jar Test. Timing data was recorded from three runs of each experiment. Bottle Carrying Test documentation also included rate of failure.

3.5 General Experimental Protocols

The study was approved by the University of Texas at Arlington Institutional Review Board and informed consent was obtained from each volunteer. One unilateral amputee performed the Jar, Blanket Folding, and Box and Blocks tests. A group of healthy volunteers (numbers specified in each experiment) performed the Jar, Blanket Folding, Box and Blocks, and Bottle Retrieval tests. In each case, the Box and Blocks test was performed using the healthy hand (in all cases, including the amputee), employing all the control modes (healthy volunteers), and using a commercial EMG based Touch Bionics robo-limb prosthetic (in the case of the amputee).

3.5.1 Experiment 1: Towel Folding Test

In the towel folding test, the subjects (all non-amputee volunteers) were asked to fold a medium sized (90 cm by 60 cm) towel under three different conditions: 1) with both healthy hands, 2) with a single, non-dominant hand, 3) with the Gesture Glove based system. The performance was timed for 3 subjects (all right handed, 2 males, 1 female, ages 25-39) performing 3 tries each.

3.5.2 Experiment 2: Box and Blocks Test

The standard Box and Blocks was modified [89] to use a single large styrofoam block (5cm x 5cm x 5cm), instead of many small blocks. The volunteers moved from

one side of a 53.7cm x 25.4cm area to another over a 16cm high divider as frequently as possible in 60 seconds. The test was performed by two groups of volunteers on two different dates, as well as by the amputee volunteer. The healthy volunteers performed the test with the dominant hand (right hand), and then with healthy limb adapter (on dominant hand) controlled using the Gesture Glove, the Mirroring Glove, and the $iPhone^{TM}$ app provided by the manufacturer of the prosthetic limb.

3.5.3 Experiment 3: Blanket Folding Test

Volunteers were timed while folding a large blanket (243cm x 243cm) on a bed. The volunteers were instructed to fold the blanket in three, and repeated the task three times. The test was performed with both healthy hands (bimanual), singlehanded, and with the healthy limb adapter controlled by the Gesture Glove and the Mirroring Glove. In addition, one amputee volunteer completed the same task three times using her EMG controlled device.

3.5.4 Experiment 4: Bottle Carrying Test

In the bottle carrying test, three healthy volunteers were instructed to carry six empty soda bottles from a kitchen counter to another table in the room, using a tray. Completion time and number of failures (drop of any bottle) were tabulated. The experiment was re-started on failure and repeated as many times as necessary to reach success. Completion times only represent successful runs of the experiment.

The tasks in experiments 2-4 were performed with two hands (bimanual), one dominant hand (single), and with the healthy limb adapter controlled by the Gesture Glove and the Mirror Glove.

3.5.5 Experiment 5: Jar Test

Five closed jars, with a plastic block in front of each, were placed in front of the volunteers who were timed while they opened the jar, placed the block inside, and closed each jar. The tasks in experiments 2-4 were performed with two hands (bimanual), one dominant hand (single), and with the healthy limb adapter controlled by the Gesture Glove and the Mirror Glove.

3.6 Experimental Results

3.6.1 Towel and Blanket Folding Tests

Three subjects performed folding tasks (towel folding and blanket folding) three times. In both cases, bimanual performance of the task with two healthy hands was the fastest (9.5 s and 27 s respectively, averaged for 3 subjects and 3 trials), followed by the use of the Gesture Glove based system (20 s and 49.69 s) and then by the single non-dominant hand (23 s and 98.9 s). The use of the prosthetic device appears to confer the largest advantage over using a single hand when the object to be manipulated is large (in this case, the blanket). Towel Folding and Blanket Folding tasks are summarized in Fig. 3.13 and Fig. 3.14, respectively. In the towel folding task, three subjects were instructed to fold a towel with two hands, the healthy non-dominant hand, and the conductive pinch glove based system. When folding a small towel, the system does not appear to confer a great advantage over using a single hand.

3.6.2 Gesture Glove vs. EMG in Blanket Folding Test

In the blanket folding test (Fig. 3.14), the Gesture Glove performed significantly faster than the commercial EMG driven prosthetic ($49.69 \pm 21.65s$ vs. 76.38 ± 17.4, ($p = 0.042$) as determined by a Mann Whitney Rank Sum test).

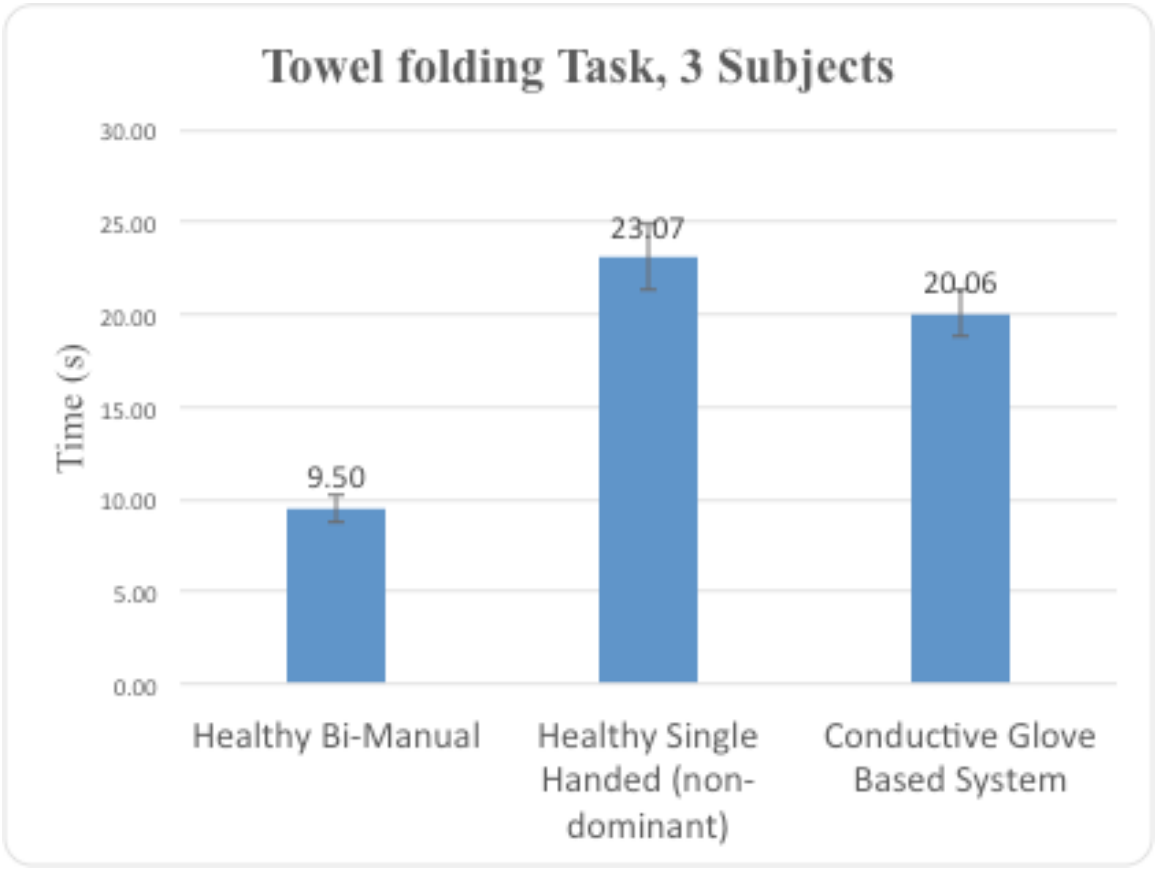

Figure 3.13 Results of the towel folding task.

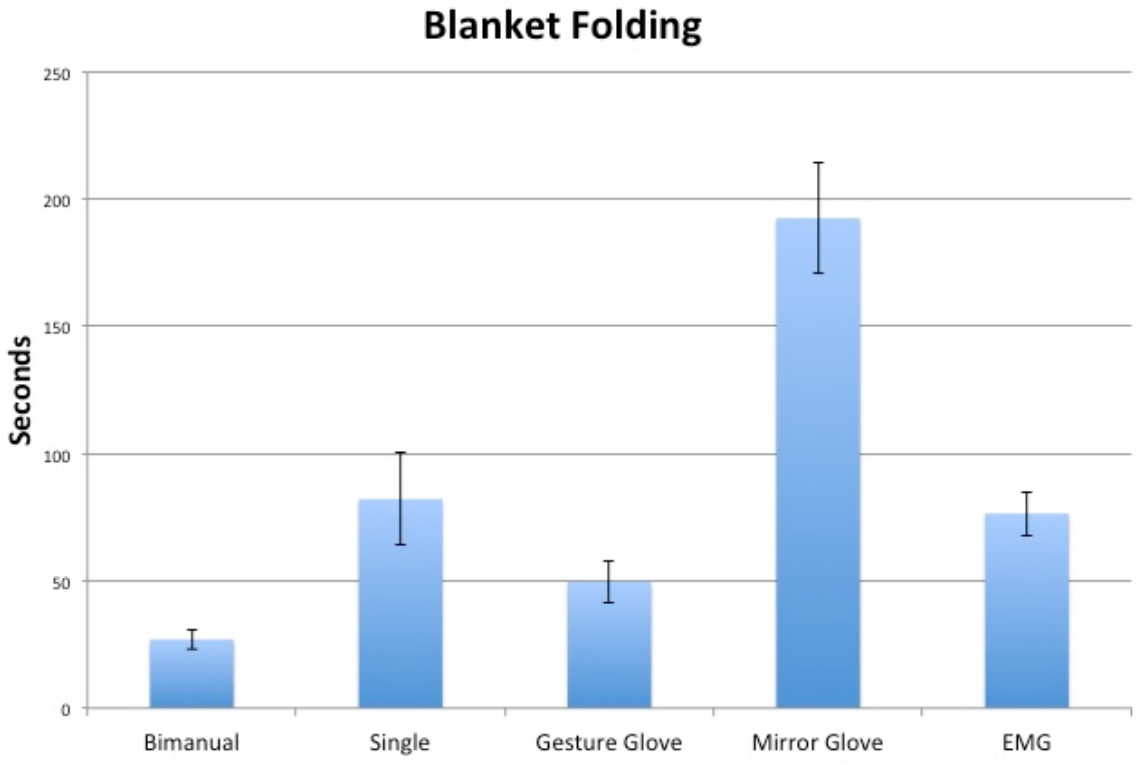

Figure 3.14 Result of the blanket folding task, where the advantage of using the Gesture Glove based prosthetic system can be seen. The subjects were asked to fold a large (243cm x 243cm) blanket. Using the Gesture Glove system is significantly faster than using a single hand ($p < 0.05$).

3.6.3 Box and Blocks Test

The results of the first instance of the Box and Blocks test are summarized in Fig. 3.15.

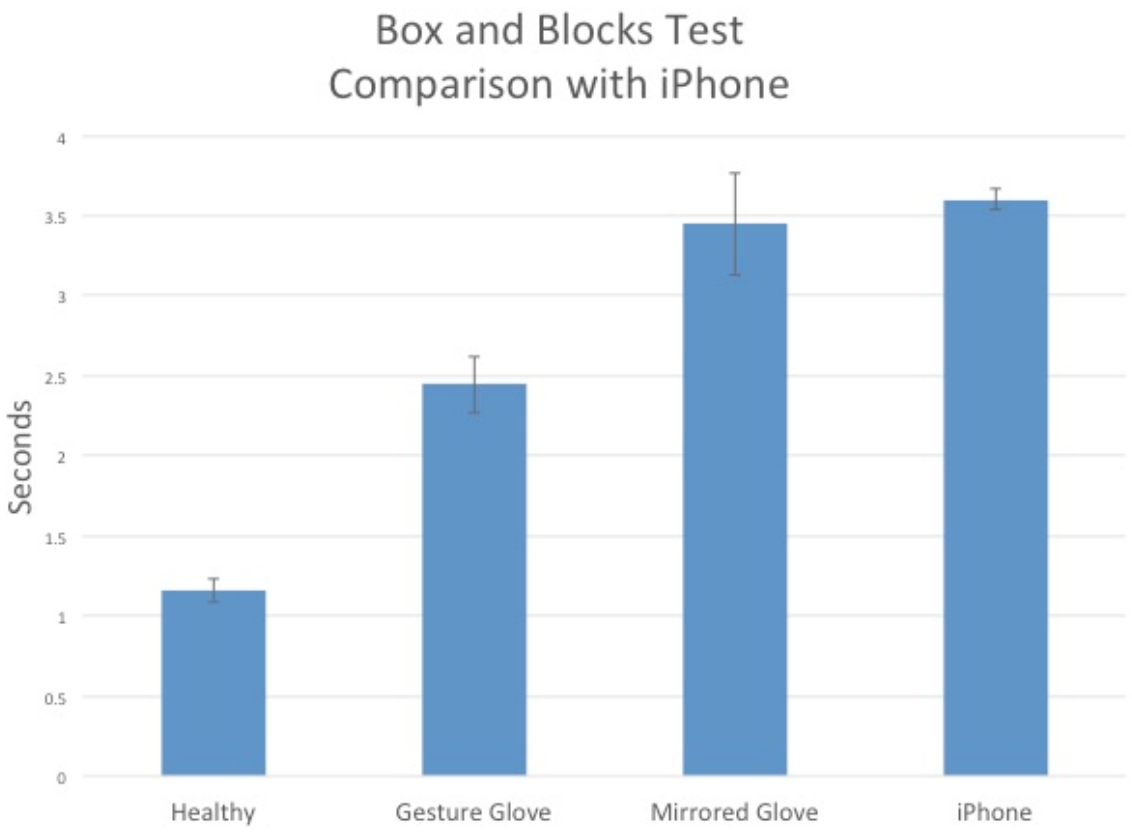

Figure 3.15 Box and Blocks Test, instance 1 (comparison with iPhone).

Manipulation with the Gesture Glove (2.44 0.35 seconds/block) was significantly faster than manipulation with the iPhone app ($3.60 \pm 0.13s$) (p¡0.001) but slower than the healthy hand ($1.16 \pm 0.14s$). The Mirrored Glove ($3.45 \pm 0.65s$) was not significantly faster or slower than the iPhone app.

The experiments below whose results are depicted in Fig. 3.16 were performed on a different group of volunteers, including one volunteer using an EMG based prosthetic system.

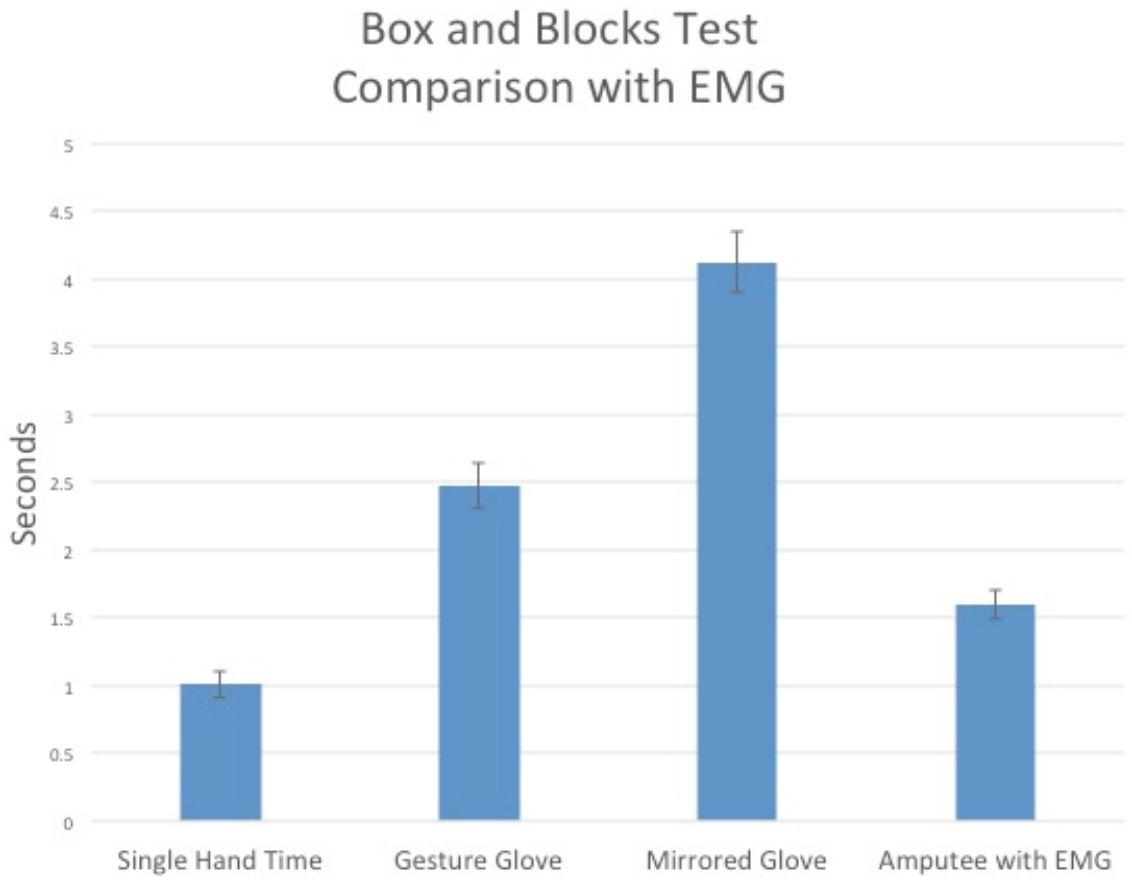

Figure 3.16 Box and Blocks Test, instance 2 (comparison with EMG).

Manipulation with the healthy hand took an average of 1.006 ± 0.19 seconds, the Gesture Glove took an average of 2.47 ± 0.33 seconds, Mirrored Glove took an average of 4.12 ± 0.45 seconds. The average time for the amputee using an EMG device was 1.60 ± 0.21 seconds. The same amputee volunteer was able to complete each block manipulation task in 0.99 ± 0.05 seconds with her healthy hand.

3.6.4 Bottle Carrying Test

For the Bottle Carrying Test in Fig. 3.17, both the amount of time taken on a successful completion and the number of failures (dropped bottles) occurring before a completed test were quantified. Healthy volunteers completed the test in 27.3 ± 6.63 seconds, with 0.22 failures on average with two hands, and in 43.11 ± 9.89 seconds with (0.83 average failures) with one hand. The Gesture Glove controlled task took $59.33 \pm 7.31s$ (1.33 average failures). Use of the Mirroring Glove reduced the time to $42 \pm 3.34s$ (0.67 average failures). Mirroring Glove was significantly faster (p¡0.001) than gesture based control, and had a significantly lower number of failures (p¡0.001) than the single-handed performance of the test.

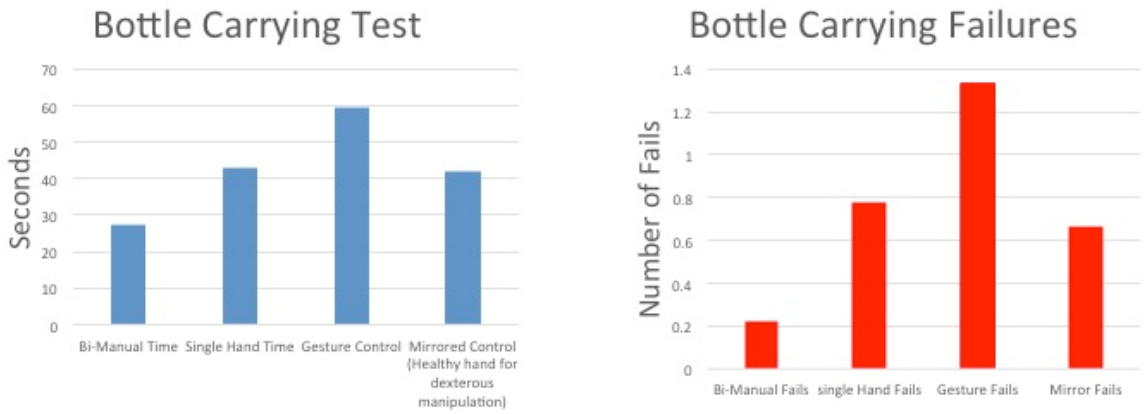

Figure 3.17 Bottle Carrying Test. Left: Time taken for successful attempts. Right: Average number of failures before the achievement of a successful attempt for each control scheme.

3.7 Discussion

Two novel healthy hand-based prosthetic control paradigms for a unilateral amputee were tested on volunteers: the Mirroring Glove and the Gesture Glove. The

preliminary results indicate that our prototype systems can be faster than EMG based control in some instances (blanket folding with gesture based control).

The Mirroring Glove and the Gesture Glove are novel intact-hand based prosthetic control interfaces offering advantages in terms of deployability, usability, and task completion durations. Inexperienced volunteers required around 20 to 30 minutes of training. We believe clinical devices can be based on these paradigms.

The fabric folding tests represent a subset of tasks in which it is useful to have two hands and to be able to execute a grasping motion (as opposed to bimanual tasks in which simply having one residual limb with no grasping ability might suffice, such as holding a jar in place while opening it). Having a powered prosthetic device appears to be helpful in fabric folding only when the piece of fabric being folded is large (such as a large blanket).

The Mirroring Glove achieved simultaneous control of 5 DOFs and allows the user to perform tasks which are naturally mirrored (such as carrying large objects, pulling a table, etc.) or use the intact hand to pose the prosthetic device before locking it in place [42]. Volunteers report that the Mirroring Glove was generally easier to use for these tasks, with the exception of blanket folding, where the Gesture Glove was superior. We hypothesize this is due to the preliminary nature of the hardware and limitations of the prosthetic device which prevent mirroring all human hand DOFs. Performance of the current device will be improved with a more sophisticated and accurate gesture recognition algorithm and the addition of thumb opposition as a DOF.

The Gesture Glove leaves the fingers free and allows the user to quickly pose the prosthetic device in pre-determined configurations using pinch gestures [90]. The main advantage is that four gestures are accessible in the time it takes to tap a finger to the palm (approximately 200 ms [91]). Additional poses can be mapped to gesture chords using multiple fingers or timing based input methods such as double-pinch or long pinch. This device consistently outperformed the other control modalities except during the Bottle Retrieval Test. Limitations include skin conductivity changes such as sweating and the touching of metal objects with both fingers and palm, which will be addressed with improved signal processing.

3.7.1 Developed Systems in the Context of UI Design

The first device developed for this work, the Gesture Glove, is essentially a zero friction system (in the sense of "friction" as used in UI design literature such as in [20]) in that the movements of the intact hand are mirrored directly. In this context, an additional point of intentional "friction" needs to added in order to allow the user to lock the hand in place both in order to hold an object and also in order to avoid unwanted movement in the robotic device. Conceptually, this is the most intuitive device to use, since every movement of the intact hand is mirrored in the prosthetic hand.

The Gesture Glove and the Fingernail Device (discussed in Chapter 4) are also low friction devices, but they require intentional actuation by the user. It is possible, however, to perform a gesture without intending to. In order to prevent this, a somewhat uncommon gesture such as "double tap" needs to be in place to place the system into and out of gesture recognition mode.

Interestingly enough, the Gesture Glove and Fingernail Device were easier to use by the volunteers, as the speed of the robotic device made it somewhat difficult to position all the digits properly.

3.8 Conclusion

Unlike EMG-based control, our prototype interfaces can be quickly learned by amputees and may be particularly suitable for unilateral transradial amputees who lack enough residual muscles to control an EMG-based prosthetic. Another possible use is complementing EMG-based control, such as putting the device in various operation modes. Future work includes making the devices faster, more accurate, less cumbersome, and validating their performance with more users, including amputees.

3.9 Future Work

Both the Mirroring Glove and the Gesture Glove hold potential to augment or replace EMG based control, but certain improvements need to be made to these devices before they are ready to be used in a clinical setting. The first such improvement is the addition of wireless capability, which is in progress.

If the Mirroring Glove is to be used with a prosthetic device which only has commands to "close" and "open" each digit (as is the case with the Touch Bionics robo-limb device used in this work), the glove firmware needs to be updated to wait until the user has indicated the desired hand pose before the command to move the digits is issued – otherwise, the robotic hand is prone to jamming due to the inability of the motors to keep up with the speed of natural finger movement (each digit on the robotic hand takes approximately 1.4 seconds to open or close, which is much slower

than what the human hand is capable of).

Another improvement which is in progress is the addition of accelerometer based lock switches, in order to prevent accidental movement of the robotic limb. Currently, only the Mirroring Glove has an external switch for locking, which is actuated by touching it to a part of the user's body (such as the chin). The ideal embodiment of this may be an accelerometer based switch which the user activates by shaking the hand on which the glove is worn. The exact embodiment of the switch can be determined by the patient's orthotist depending on the patient's disability and needs.

An important concern expressed by unilateral amputees is that they do not wish to encumber their healthy limb with any more hardware than is absolutely necessary. In light of this, any healthy limb based control device needs to be as lightweight and unobtrusive as possible. This concern lead to the development of the Fingernail Device, which is a lower encumbrance device worn on the fingernails. The device has similar functionality to the Gesture Glove. The advantage of this device is that it can be deployed in a wireless form which is also devoid of any haptic encumbrance at the finger pads. The Fingernail Device and its construction, testing, and operation are covered in the next chapter.

Work is also under way to repeat the Activities of Daily Living tests for the Fingernail Device.

CHAPTER 4

DEVELOPMENT AND TESTING OF A LIGHTWEIGHT FINGERNAIL WORN SENSOR

4.1 Motivation

The previous sections have explored the concept of using a Mirroring Glove and a Gesture Glove to quickly pose a prosthetic device using the intact hand. This mode of control allows robotic prosthetic devices to be used by patients who are unable to generate proper EMG signals, and provides an alternative or a complement to EMG based control for those who can.

In order to be deployable and usable by patients in the performance of Activities of Daily Living (ADLs), a healthy limb based system needs to be free of haptic encumbrances and leave the fingers (and especially the fingertips) of the healthy hand free.

Several attempts have been made to develop and deploy wearable devices with low encumbrance. We present a device based on light transmittance through two fingers forming a pinch gesture from an LED emitter placed on each fingernail to a detector on the thumbnail. This device exploits the phenomenon of increased light transmittance which occurs when the two fingers make contact, temporarily forming a light guide. Our system has the advantage of requiring a very small amount of electronics on each fingernail mounted device, making the entire system easy to construct

and deploy.

Pinch gestures are intuitive gestures which can be performed by touching the fingers of the hand to other fingers or the palm of the hand [46]. Pinch gesture tracking gloves are commercially available but require the use of a glove on the healthy hand, which ultimately limits manipulation ability with that hand. These systems typically use conductive fabric and the completion of a circuit by touching various parts of the glove together. Other gesture tracking systems require the use of a camera or other 3D tracking system. Kim et al. [92] have proposed a gesture tracker using a wrist worn camera which obviates the need for a glove, but introduces occlusion errors. Nonetheless, they have demonstrated the ability to control various devices by performing both simple and complex gestures with the hand. Other types of gesture systems such as the Myo [52] and the GEST [93] system (https://gest.co) rely on EMG bracelets or accelerometers linked to a lightweight bracelet/glove system. The Myo is an EMG based bracelet which also uses accelerometers to allow the performance of various gestures defined by the manufacturer to allow for the control of various types of applications such as mp3 players, slide presentation software (PowerPoint), or computer games. The Myo device has also been used to control a prosthetic device [52]. The GEST [93] system consists of a bracelet which is connected to accelerometer-containing rings worn on each finger in order to recognize various gestures. Mascaro et al. [94] have created a fingernail mounted system which works by using miniature LEDs and photodiodes in order to detect color changes in the fingernails in response to pressure applied to the fingertips.

The basis of our device is a compact nail worn light transmitter-receiver pair that can detect contact events. We describe our initial prototype consisting of fingernail sensors interfaced to a microcontroller, demonstrating that the proposed method-

ology can ultimately lead to an extremely low cost and completely untethered device. Even though the proof of concept prototype presented is tethered to a microcontroller, the system we propose does not have any inherent need to be tethered and the transmitters on the fingernails can be built to be untethered without requiring any communication with the rest of the system except the light they emit/detect.

4.2 Physical Design of Current System

Fingernail mounted sensor holders for an LED and a photodiode were modeled in SolidWorks and 3D printed using a filament extruder based 3D printer. A high powered (280 mW) far red (730 nm) LED (GF CSHPM1.24-3S4S-1, OSRAM Opto Semiconductors) and an infrared phototransistor (BPY 62-4, OSRAM Opto Semiconductors) were fit into the sensor holder with moldable plastic (InstaMorph, www.instamorph.com). Both the LED and the sensor on each sensor holder were soldered to thin wires. The wires were connected via DuPont connectors to a wrist worn Arduino board. The microcontroller board was constructed out of a Velcro wristband, an Arduino Genuino microcontroller board, and a custom Arduino shield with circuitry to drive the LEDs and read the input from the photodiodes as depicted in Fig. 4.2. The prototype system created using one thumb sensor and two fingernail sensors is depicted in Fig. 4.5.

The described basic system is driven by the circuit illustrated in Fig. 4.1. The phototransistor illustrated in the circuit is mounted on the thumb and the LEDs are mounted on the fingernail based emitters.

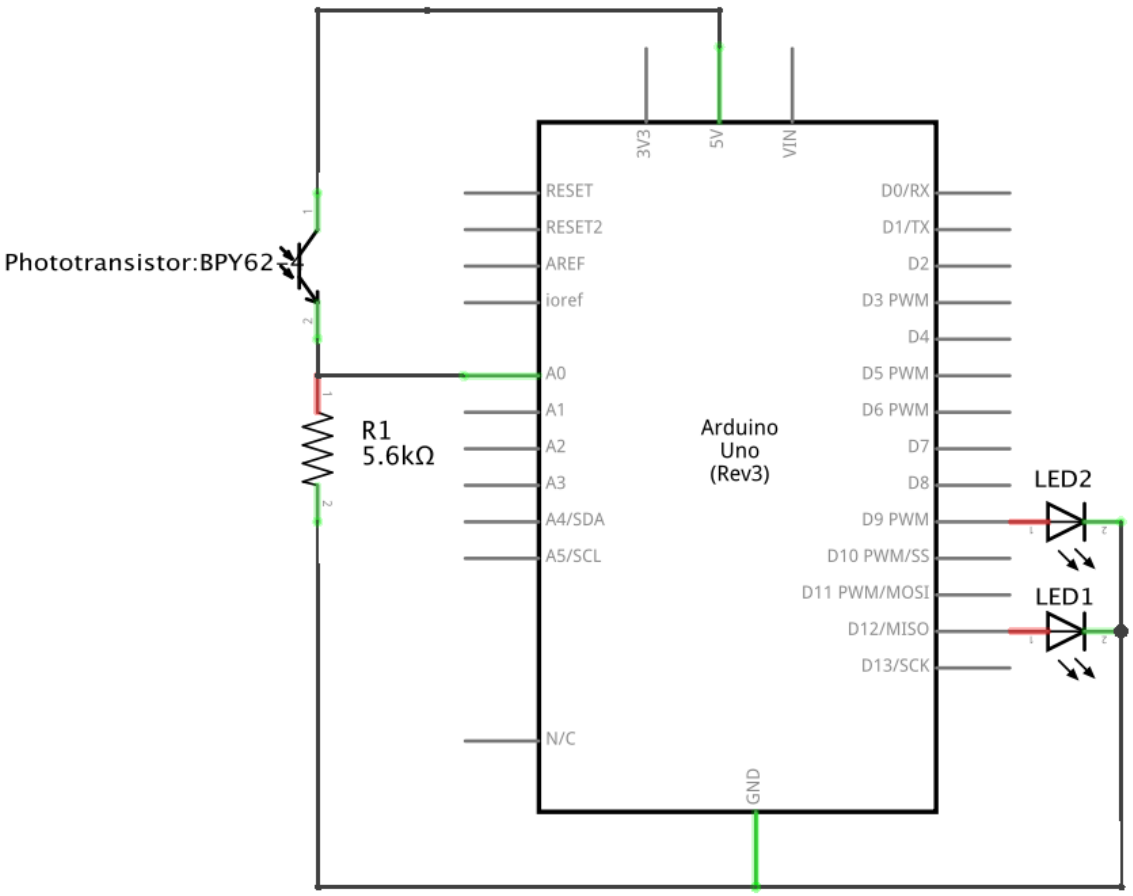

Figure 4.1 Basic circuit for proof of concept implementation of Fingernail Device.

While the device functions as desired using the 280 mW LED, it is desirable to reduce the power being used for various reasons including increased battery life (for an untethered emitter) and safety.

It is possible to improve the performance of the system and detect light going through two fingers by using the circuit in Fig. 4.3. The transistor in this circuit allows more current to be driven through the LED. If the circuit in Fig. 4.3 is used with an Arduino device, the Arduino should be externally powered in order to

avoid noise from drawing too much current through the USB port of the computer. Another alternative circuit for amplifying the light transmitted through tissue is the transimpedance amplifier circuit given in Fig. 4.4. This circuit also utilizes an I^2C based 16 bit analog to digital converter board with programmable gain (ADS1115, Adafruit Industries, USA).

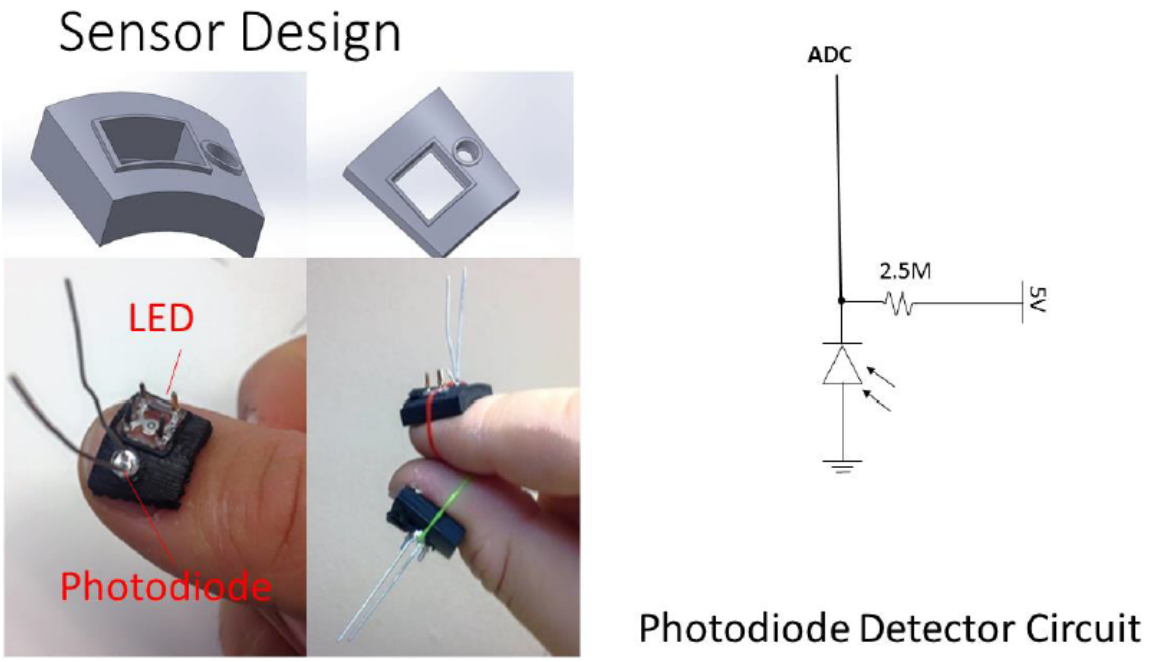

Figure 4.2 Left: Sensor holder design and implementation Right: Photodiode based detector circuit.

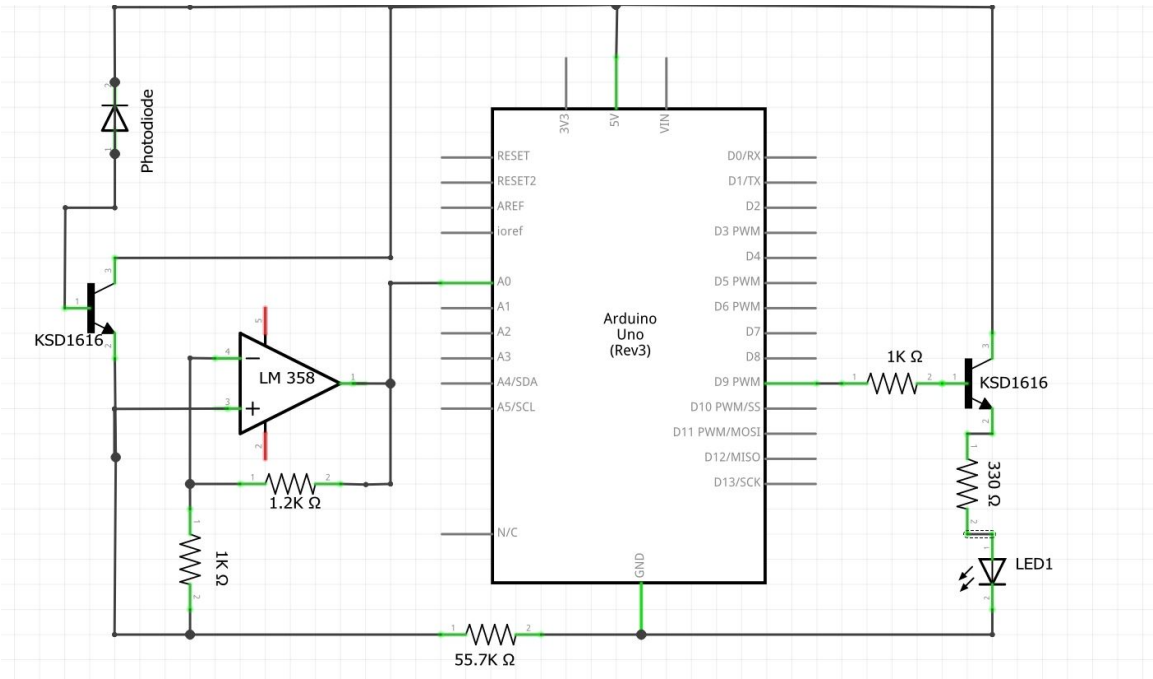

Figure 4.3 Bipolar Junction Transistor and OpAmp based circuit to amplify light through two fingers. One flashing LED is shown on the right. Circuit diagram from Mr. Brandon Young.

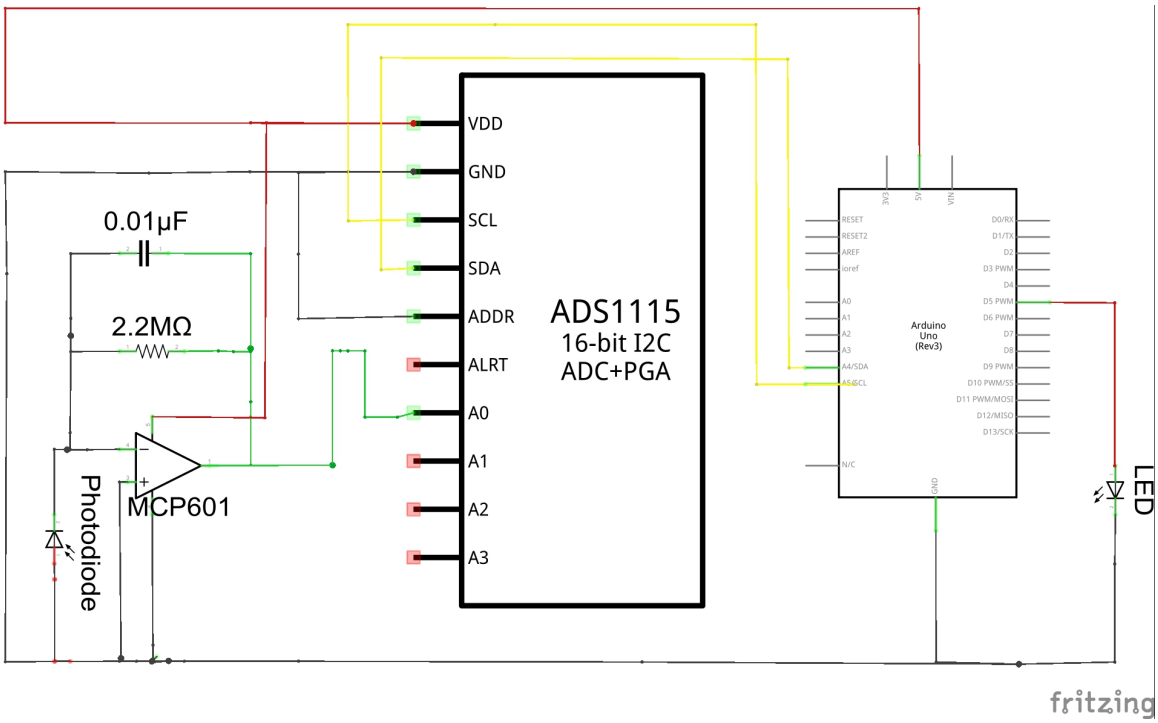

Figure 4.4 Transimpedance amplifier used with a 16 Bit Analog to Digital converter device which as its own programmable gain. Circuit diagram from Mr. Simranjit Ahluwalia.

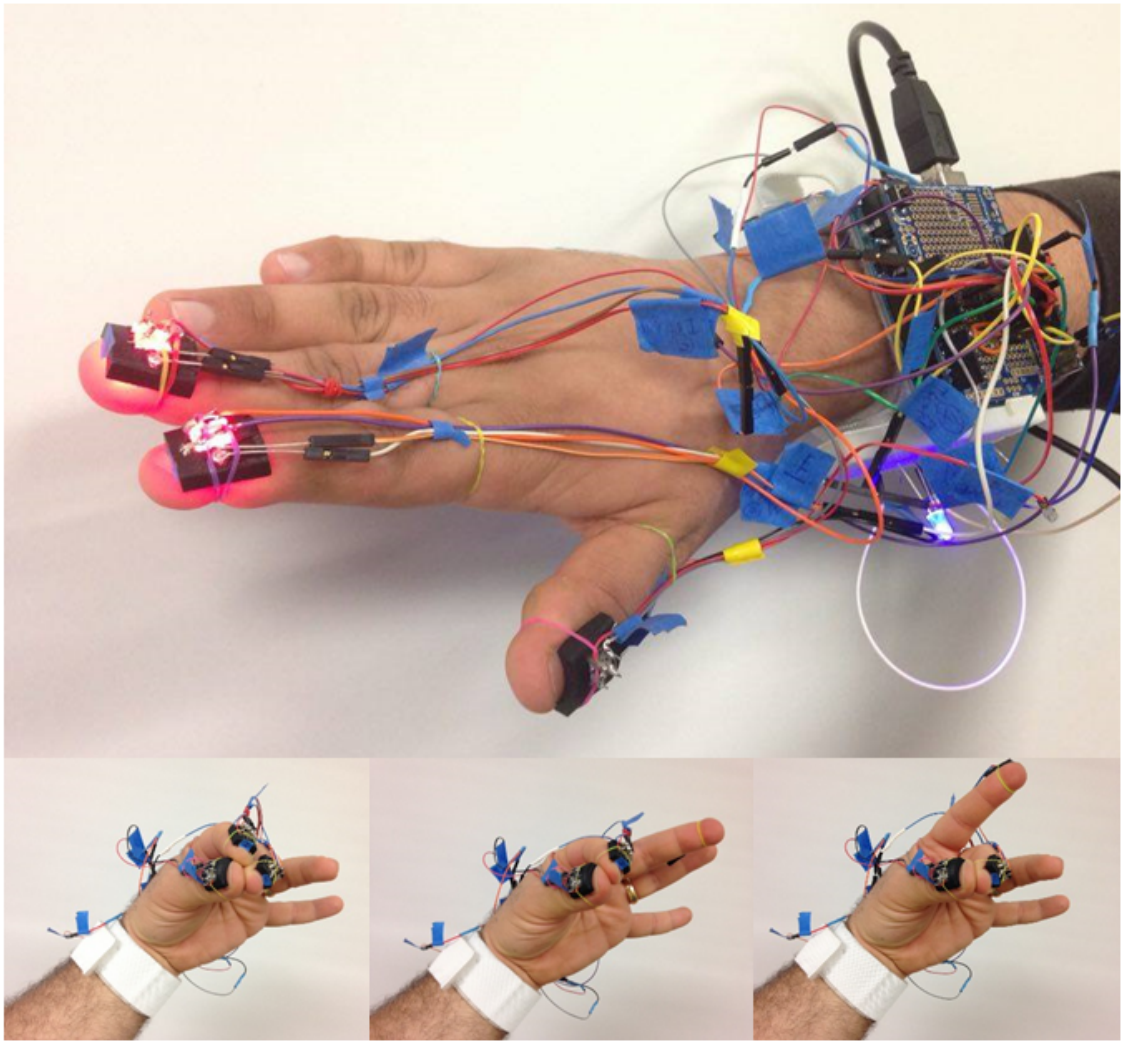

Figure 4.5 Proof of Concept system with a wrist-worn microcontroller (top) and three pinch gestures (bottom).

Certain wavelengths of light travel through tissue [95]. The phenomenon is further exploited by the use of enhanced light transmission when the two fingers touch, thus allowing light directed at a fingernail to be detected at a sensor aimed at the thumbnail. This phenomenon is illustrated in Fig. 4.6 and Fig. 4.7.

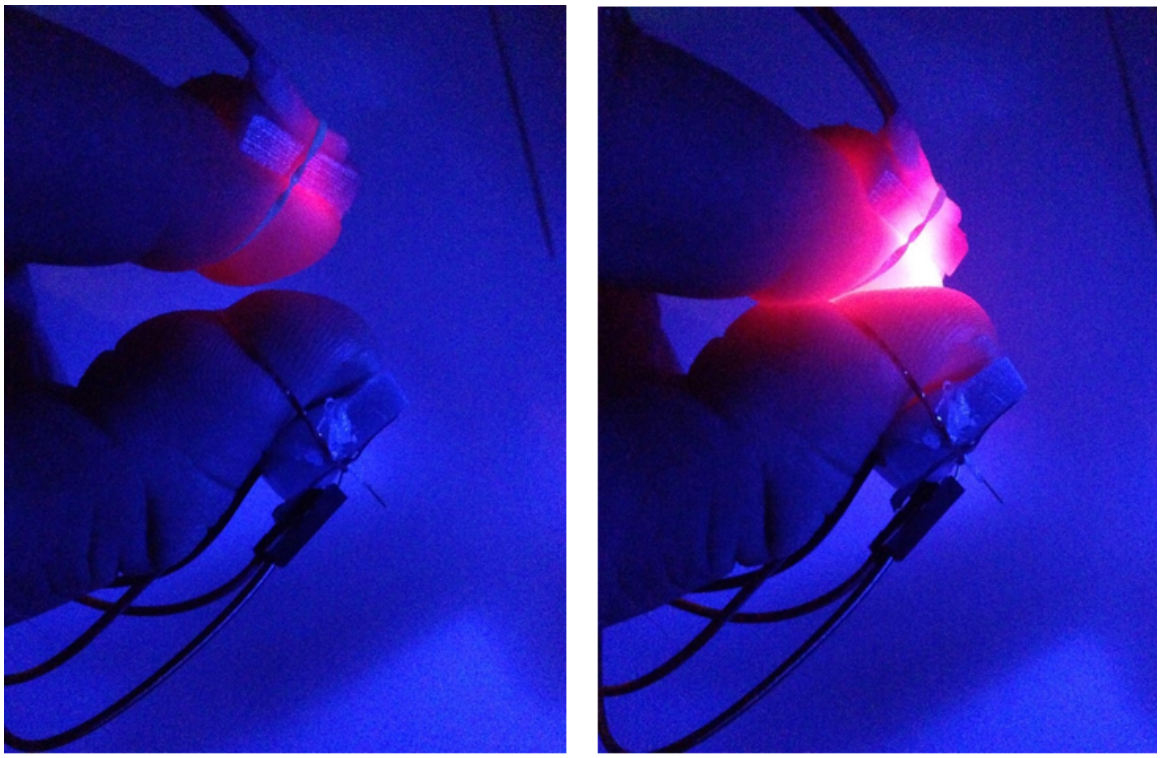

Figure 4.6 Light transmission through connected fingers when one finger is illuminated.

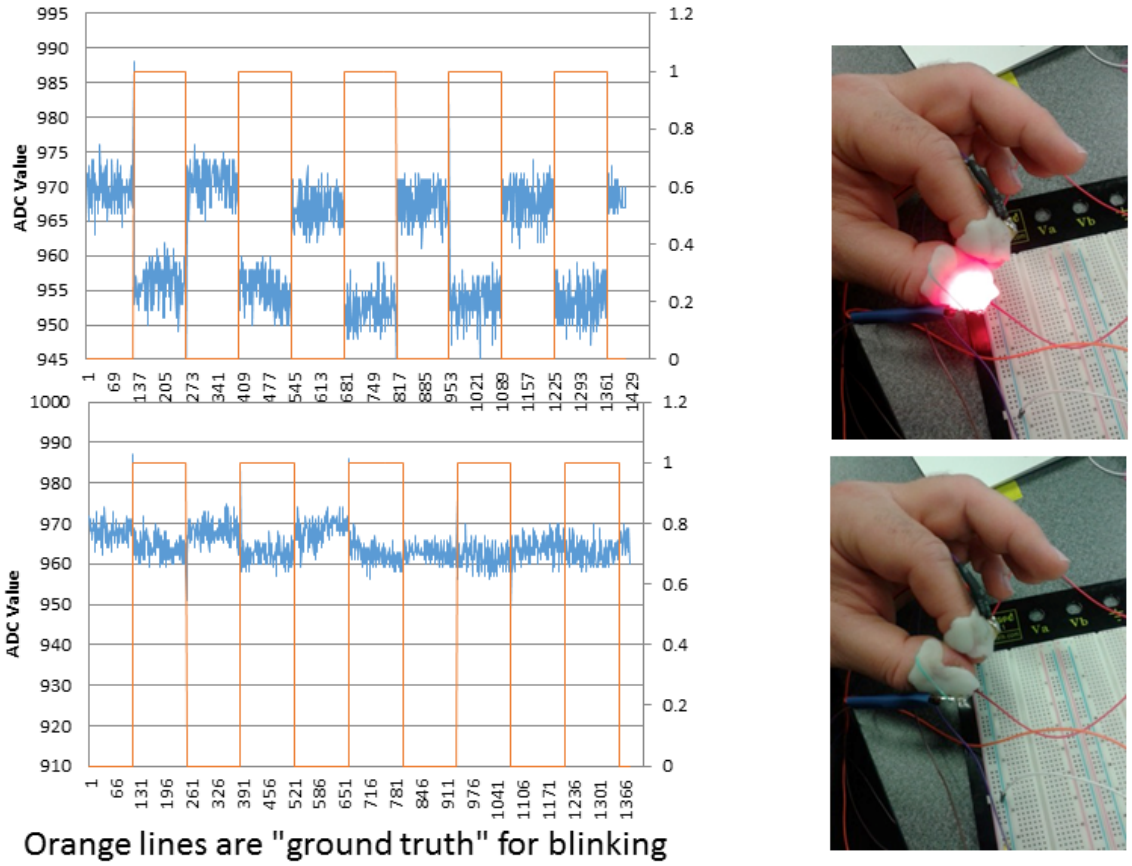

Figure 4.7 Light detected on a fingernail after traversing through two fingers in physical contact (top), and light detected on the same fingernail after traversing through two fingers with a small air gap between them.

4.3 Detection Methods

With the system as designed, several detection methods are possible in order to detect a contact event. These include Time Based Detection, Light Interrogator Based Detection, and Fast Fourier Transform (FFT) Based Detection. Time Based Detection is the simplest method and flashes LEDs mounted on different fingernails at different times. It does, however, require that all the LEDs be tethered to the microcontroller. Light Interrogator Based Detection allows the LEDs on the fingernails

to be untethered and in low power mode until they receive an "interrogation pulse" from an LED mounted on the thumbnail. This event will only happen during the performance of a gesture, and thus has the advantage of preserving the batteries on the fingernail based emitters. FFT based detection also allows the fingernail based emitters to be untethered. In this method, the LEDs on each fingernail are flashed at different frequencies and an FFT based detection algorithm is used to distinguish between the fingers. FFT Based Detection has the advantage of being able to reject noise and allows the emitter power to be limited. In future implementations, the Light Interrogator Based Detection scheme should be combined with FFT based detection in order to reject noise and preserve battery life on the fingernail mounted emitters.

4.3.1 Time Based Detection

In this detection scheme, the system described in Fig. 4.2 and Fig. 4.5 is used to detect touch events.

For this detection method, each detection cycle is broken into detection windows of length $window\_duration$ $e.g., 200ms$. Each finger is assigned a characteristic response time r1,r2,r3,r4. These times are chosen to be smaller than $window\_duration$ ($e.g., 25ms, 40ms, 57ms, 75ms, etc.$) but larger than $pulse\_duration$ ($e.g., 5ms$).

During each detection window, the LEDs are flashed in succession. Simultaneously, the microcontroller reads the light intensity on the photodiode and looks for peak intensities corresponding to the response times ($r1$, $r2$, etc.). Detection of such a peak is counted as a touch event. While there is always a chance of a spurious activation of the sensor, reducing the $pulse\_duration$ helps prevent this, since the

system detects the time of the brightest signal within *window_duration*—a smaller *pulse_duration*/*window_duration* ratio is therefore preferred. In order to make the system even more robust against noise, a *detection_window* (e.g., 3) variable is defined, which will require a number of successive detection events to occur before the system will report a completed gesture. Although light intensity decays exponentially with distance according to Beer-Lambert law [58], enhancement of light transmission by the two fingers coming into contact provided a reliable touch signal in 4.7.

4.3.2 Light Interrogator Based Detection

The system depicted in Fig. 4.2 and Fig. 4.5 was modified slightly to allow the sensors on each finger to be controlled by their own untethered microcontroller. The light interrogator is similar in operation to the time based detection system except that an interrogation pulse is sent from the thumb to the sensors mounted on each fingernail. If the sensor on the fingernail detects a light peak, it responds after a characteristic response time (r1 for index finger, r2 for middle finger, r3 for ring finger, r4 for little finger. Nominal values of 25, 40, 57, and 75 ms can be chosen). The sensor on the thumb detects the response and registers a touch event if one or more peaks are detected within response times corresponding to a finger. The light interrogator system has the added advantage of being more robust against noise than the time based detection system described above, since it is very unlikely that a spurious light peak from the environment will coincide with an interrogation pulse. The operation of this system is depicted graphically in Fig. 4.8. The light interrogator system is depicted in Fig. 4.8. This system has the advantage of allowing fingernail based sensors to be completely untethered and manufactured very cheaply, as all they need to do is to respond to a light pulse. This furthermore increases battery life, since

the fingernail mounted sensors only have to respond to a pulse if the fingers are in contact. To date, a prototype Light Interrogator Based Detection built using the detection scheme described here has been demonstrated to work with a $window_duration$ as short as 50 ms, which should be fast enough to enable additional gestures such as "double-tap" or "triple-tap," similar to a double-click on a computer mouse.

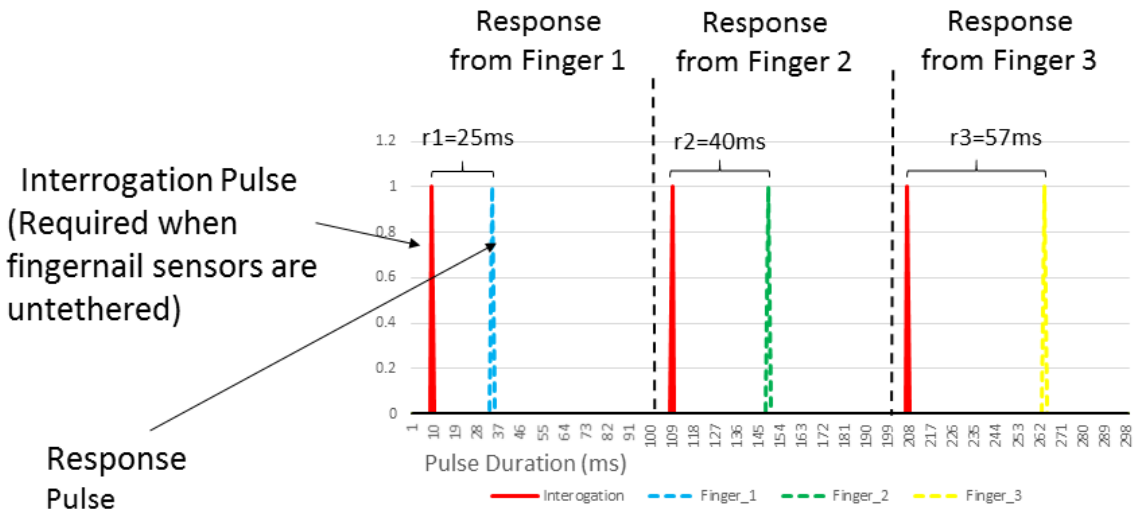

Figure 4.8 Light Interrogator Based Detection.

4.3.3 FFT Based Detection

The time domain based systems above are prone to noise and make the assumption that the signal from the LED will be the brightest light pulse detected by the photodiode during the detection window. This makes it difficult to detect light traveling from a fingernail to the thumbnail without using a very bright LED. An alternative approach using the Fast Fourier Transform (FFT) involves placing LEDs flashing in different frequencies on each fingernail, and detecting the light traveling

through both fingers as it exits through the thumbnail when the pinch gesture is performed. This is a common use of the FFT and has been used in various contexts including [96]. This greatly simplifies the design of the fingernail mounted devices, as no microcontroller is needed on the four fingernails. The FFT based approach has the advantage of being more robust against noise and requiring lower power and smaller LEDs, since the system no longer needs to detect an absolute brightness peak. MATLAB code (frequency_detector) was written to take input of a vector of time domain data sampled from the photodiode, and return a vector containing (time, frequency) pairs where the frequency represents maximum frequency within a short window of time. This was accomplished through a sliding window approach, where the window length was 20 samples. The frequency_detector code also eliminates all frequencies outside the desired frequency range. Output from simulated runs of the frequency_detector code is presented in Fig. 4.9

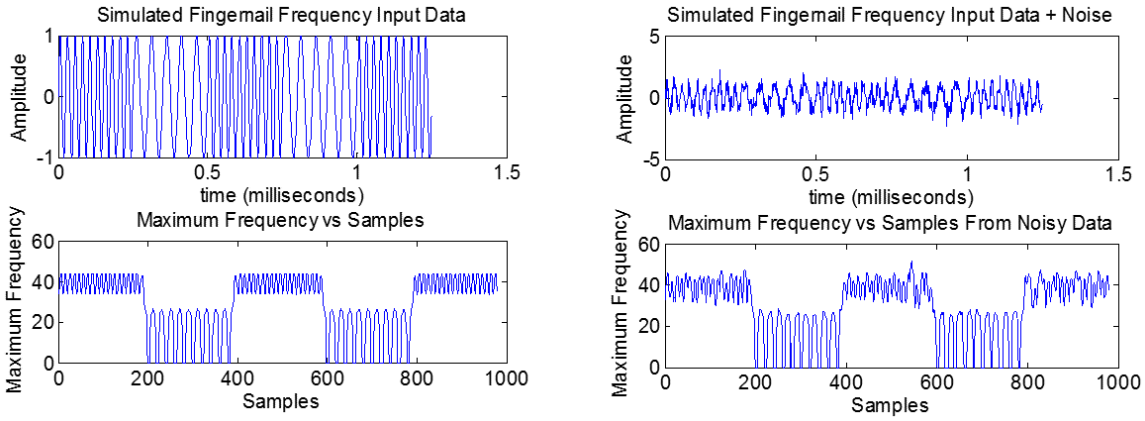

Figure 4.9 FFT based detection simulation with and without noise. Left: output of frequency_detector() for simulated data, no noise. F1= 40 Hz and F2= 20 Hz. Right: output of frequency_detector() for simulated data, + noise 60 Hz + random noise. F1= 40 Hz and F2= 20Hz.

In order to test the frequency_detector code and pick the most optimal frequency pairs which can be detected by the Arduino microcontroller board, simulations were run on different frequency pairs ranging from 10 to 400 Hz (the maximum throughput of the Arduino microcontroller board streaming data over a serial connection to a PC was found to be 800 samples/sec at 115,200 kbaud). In order to enhance the accuracy of our frequency prediction, both random and 60 Hz noise were added. The most optimal frequencies were chosen based on RMS distance calculations between normalized ground truth and result vectors Fig. 4.10.

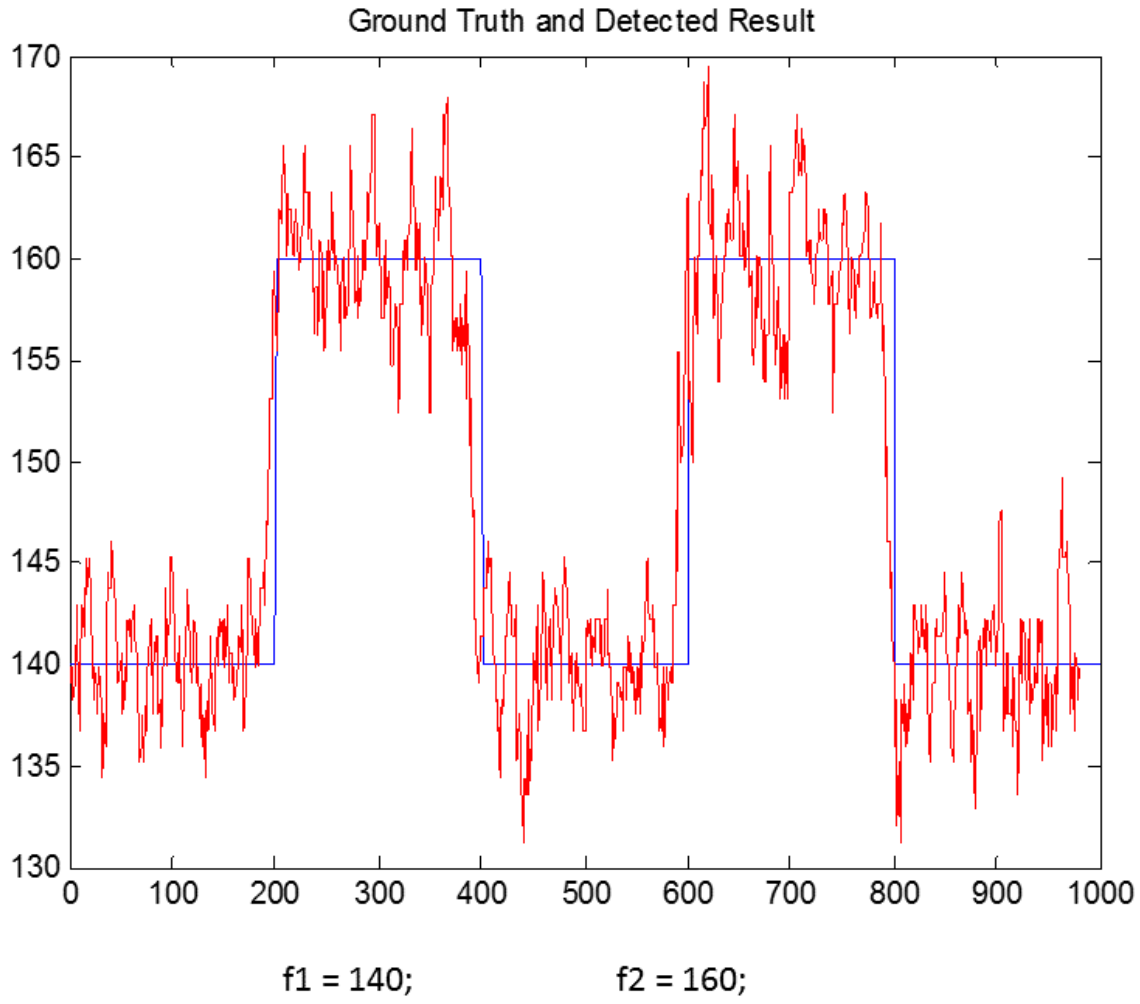

Figure 4.10 Ground Truth compared with Detected Result.

4.4 Experiments Performed

The experiments described in this section were performed by three volunteers: One healthy 23 year old right handed male volunteer, one healthy 41 year old right handed male volunteer, and one 32 year old left handed female amputee with a missing right hand. The amputee volunteer performed the Box and Blocks, Jar, and Blanket Folding tests with her own EMG-controlled device. The healthy volunteers performed

all experiments with the Fingernail Device as well as the Mirroring Glove and the Gesture Glove described in the previous chapter, in order to compare the devices under the same experimental conditions.

4.4.1 Box and Blocks Test

The volunteers wore the Fingernail Device on the non-dominant hand and the prosthetic device on the dominant hand using the Healthy Limb Adapter described in earlier sections. The device was programmed to map three different gestures on the left hand (index finger to thumbpad, middle finger to thumbpad, ring finger to thumbpad) to three grasp patterns on the prosthetic device (index finger to thumb pinch, lateral grasp, and open all fingers). The setup with the prosthetic device and the fingernail device can be seen in Fig. 4.11.

One 32 year old female amputee volunteer using her own EMG controlled prosthetic device also performed the same experiment three times.

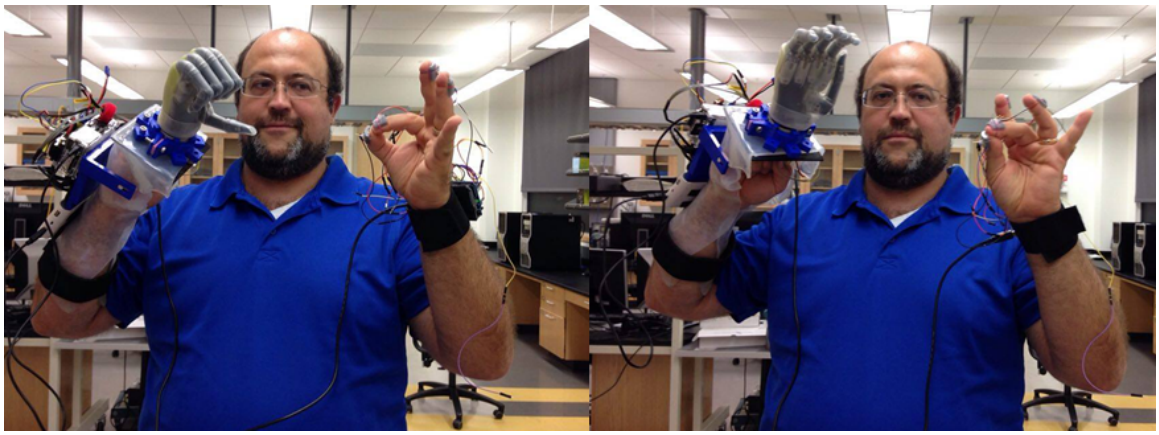

Figure 4.11 Two separate gestures performed by the Fingernail Device resulting in two different prosthetic device configurations.

A modified Box and Blocks test [89] was performed in which the volunteers moved a 5cm x 5cm x 5cm Styrofoam block from one side of a 53.7cm x 25.4cm area to another over a 16cm high divider as frequently as possible in 60 seconds. The number of movements per the time period was tabulated and converted to seconds per block. The healthy volunteers also repeated the experiment with their non-dominant hands.

4.4.2 Jar Test

The volunteers wearing the device shown in Fig. 4.11 were instructed to open 5 jars, place an object inside them, and close the jars. The total time taken to perform this experiment over three runs of the experiment was recorded.

One 32 year old female amputee volunteer using her own EMG controlled prosthetic device also performed the same experiment three times.

The healthy volunteers also repeated the experiment using two hands without a device, as well as with the Gesture Glove and the Mirroring Glove described in the previous chapters.

4.4.3 Blanket Folding Test

The volunteers wearing the device shown in Fig. 4.11 were instructed to fold a randomly placed large 243 cm x 243 cm blanket four times. The total time taken to perform this experiment over three runs of the experiment was recorded.

The healthy volunteers also repeated the experiment using two hands without a device, as well as with the Gesture Glove and the Mirroring Glove described in previous chapters.

4.4.4 Tray Carrying Test

The volunteers wearing the device shown in Fig. 4.11 were instructed to place five filled 355 mL beverage cans on a tray and carry them a distance of 5 meters to a different table. The time taken to perform this experiment over three runs of the experiment was recorded.

The volunteers also repeated the experiment using two hands without a device, as well as with the Gesture Glove and the Mirroring Glove described in the previous chapters.

4.4.5 Tissue Characterization Experiment

In order to pick optimal design parameters for the system, light attenuation using existing hardware was characterized in human tissue using one volunteer. The wavelength of the LED being used in the system (30-01SURC, Everlight Americas Inc., Carrollton, TX) was measured by a USB4000 spectrometer (Ocean Optics, Dunedin, FL). The LED intensity was measured by a PM100D optical power and energy meter (THORLAB, Newton, NJ). Two additional measurements were taken with one and two fingers (with one stacked on top of the other) in the optical path of the energy meter. Furthermore, the distance from the index fingernail to the finger pad, and the total distance from the index fingernail to the thumb fingernail while performing a pinch gesture was measured with digital calipers.

The LEDs wavelength was found to be 639 nanometers (this agrees with the specification by the manufacturer) and the power was measured to be 1248.3 microwatts. Since the sensor size and LED distance remained the same throughout the experiment, the measured power is assumed to be proportional to the LED intensity. The intensity of light was measured after passing through one and two fingers. The

distance of both one and two finger was measured in cm. The depth of the index finger was measured to be 1.182 cm, and the total path traveled by light passing through both the index finger and the thumb was measured to be 2.003 cm. Two separate values of the attenuation coefficient μ were calculated according to the Beer-Lambert law (Equation 2.1) and averaged.

As a result of the tissue characterization experiment, the 639 nm LED was replaced with an LED with a higher wavelength (730 nm) for the final implementation of the system.

4.4.6 Gesture Performance Experiment

In order to characterize the performance of our system, one healthy volunteer with no dexterity problems wore the system on his non-dominant hand as depicted in Fig. 4.13. In addition to the circuitry required for the fingernail sensors, two LEDs (yellow and blue) were mounted on the wrist of the subject. These LEDs were flashed in succession to indicate one of three possible actions in Fig. 4.12 in 1.2 second intervals.

Gesture Prompt Protocol	
Action	Response
0 (no LED)	No gesture
1 (Yellow LED)	Touch pad of index finger to thumbnail
2 (Blue LED)	Touch pad to middle finger to thumbnail

Figure 4.12 Gesture Prompt Protocol. Action 0 (no LED): No gesture. Action 1 (Yellow LED): Touch pad of index finger to thumbnail. Action 2 (Blue LED): Touch pad of a middle finger to thumbnail. A total of 10 touch events were recorded along with the time data and a value indicating which gesture was being prompted.

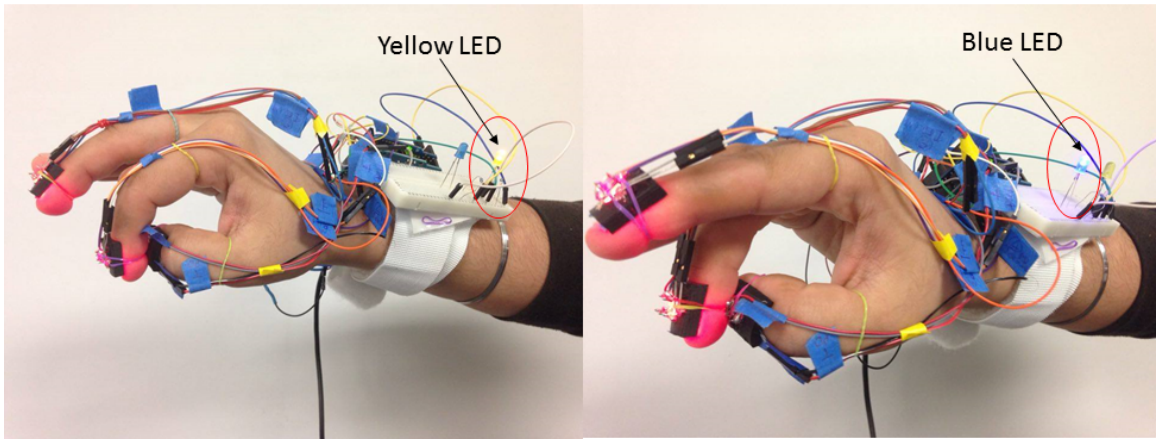

Figure 4.13 Performance Experiment Setup.

4.4.7 Single Digit Response Time Experiment

In order to ascertain the ideal response time of our pinch gesture system, a gesture timing experiment was performed using a small momentary pushbutton worn on either the thumbpad or the thumbnail. The purpose of this experiment is to determine which gestures are the fastest for subjects to perform, and to create ideal times with which to compare the current system.

Two male and two female right handed volunteers ages 22-47 with no manual dexterity problems were recruited and signed informed consent forms approved by the Institutional Review Board at the University of Texas at Arlington. Each subject was able to perform all 16 possible gestures (4 thumbnail gestures on the left hand, 4 thumbpad gestures on the left hand, 4 thumbnail gestures on the right hand, 4 thumbpad gestures on the right hand).

A small momentary pushbutton sewn onto a small band of $Velcro^{TM}$ was affixed to the subject's thumb, covering either the thumbpad (for thumb pad gestures) or the thumbnail (for Thumbnail Gestures). The subject was asked to observe an LED and press the button with a specified digit when the LED turned on. The LED timing was randomly varied between 0.5 to 2.0 seconds. For each digit, the subject repeated the experiment 10 times in a row before moving on to the next digit. In total, each subject performed three thumbnail gesture experiment sets and three thumbpad gesture experiment sets on each hand. The time taken by the subject for each prompted gesture was recorded.

4.4.8 FFT Based Detection through Two Fingers

The setup depicted in Fig. 4.13 was utilized with the prompting LEDs. The index finger LED was flashed at 41 Hz and the middle finger LED was flashed at 75 Hz with pulse duration of 10 ms. The current supplied to the LED during the pulse was 20 mA. These LEDs flashed in succession to indicate one of the three possible actions in Fig. 4.12 in 1.2 second intervals. The FFT MATLAB Code in Appendix C.3 was utilized to process the data.

4.5 Experimental Results

In this section, the experimental results obtained from the two healthy volunteers using the Fingernail Device are compared with the results from the amputee volunteer using her own EMG-controlled prosthetic device as well as the healthy volunteers using the Mirroring Glove and the Gesture Glove discussed in the previous sections under the same experimental conditions. The results are summarized in Table 4.1.

Table 4.1 Fingernail Device performance Compared with Other Devices. The experiment was performed on two healthy volunteers and one amputee. All values are given in seconds ± standard deviation.

	L Hand	Bimanual	Mirror	Gesture	Fingernail	EMG
Box & Blocks	0.93 ± 0.06		4.0 ± 0.90	4.0 ± 0.35	2.96 ± 0.51	1.6 ± 0.21
Jar		22.17 ± 6.21	94.17 ± 25.44	59.00 ± 19.13	1.72 ± 0.86	10.66 ± 5.33
Blanket	80 ± 10.94	28.83 ± 2.99	91.50 ± 17.81	77.0 ± 24.03	76.17 ± 21.88	76.47 ± 14.41
Tray		8.17 ± 2.48	26.83 ± 5.71	16.50 ± 2.51	25.67 ± 5.24	

4.5.1 Box and Blocks Experiment Results

The volunteers were able to perform the Box and Blocks test in an average of 2.96 ± 0.51 seconds per block. This can be compared to 1.60 ± 0.21 seconds per block which was measured from an amputee using her own device to perform the same test. This result is also depicted in Fig. 4.14.

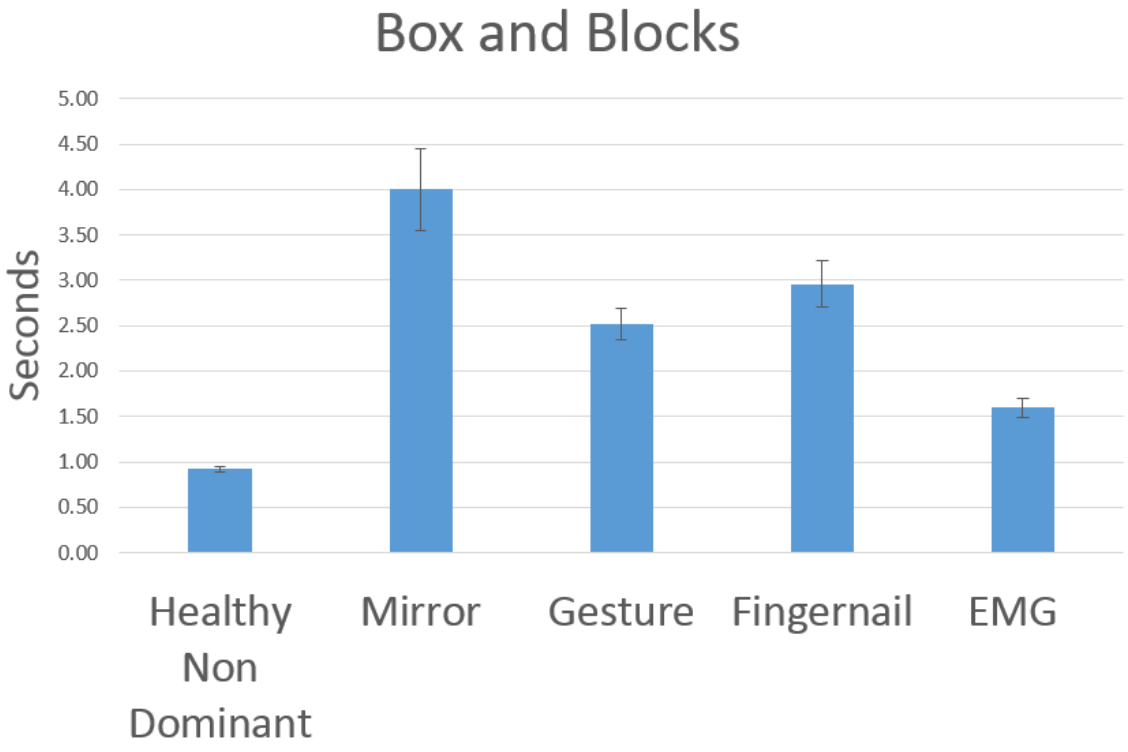

Figure 4.14 Box and Blocks Experiment Results for Fingernail Device, EMG, and the other devices discussed earlier.

4.5.2 Jar Test Results

The volunteers were able to perform the Jar Test in an average of 80.83 ± 1.72 seconds per five jars. The amputee volunteer using an EMG controlled device took an average of 83.88 ± 10.66 seconds to perform the same task, as depicted in Fig. 4.15.

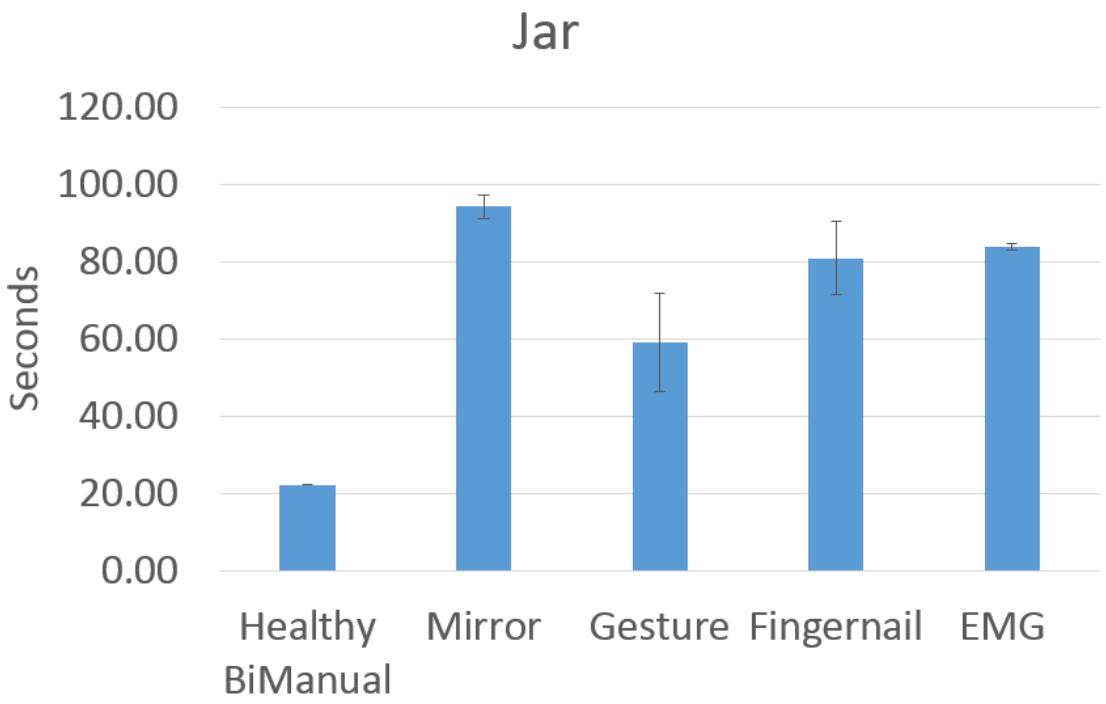

Figure 4.15 Jar Test Results for Fingernail Device, EMG, and the other devices discussed earlier.

4.5.3 Blanket Folding Test Results

The volunteers were able to perform the Blanket Folding Test in an average of 76.17 ± 21.88 seconds. The amputee volunteer using an EMG controlled device took an average of 76.47 ± 17.41 seconds to perform the same task, as depicted in Fig. 4.16.

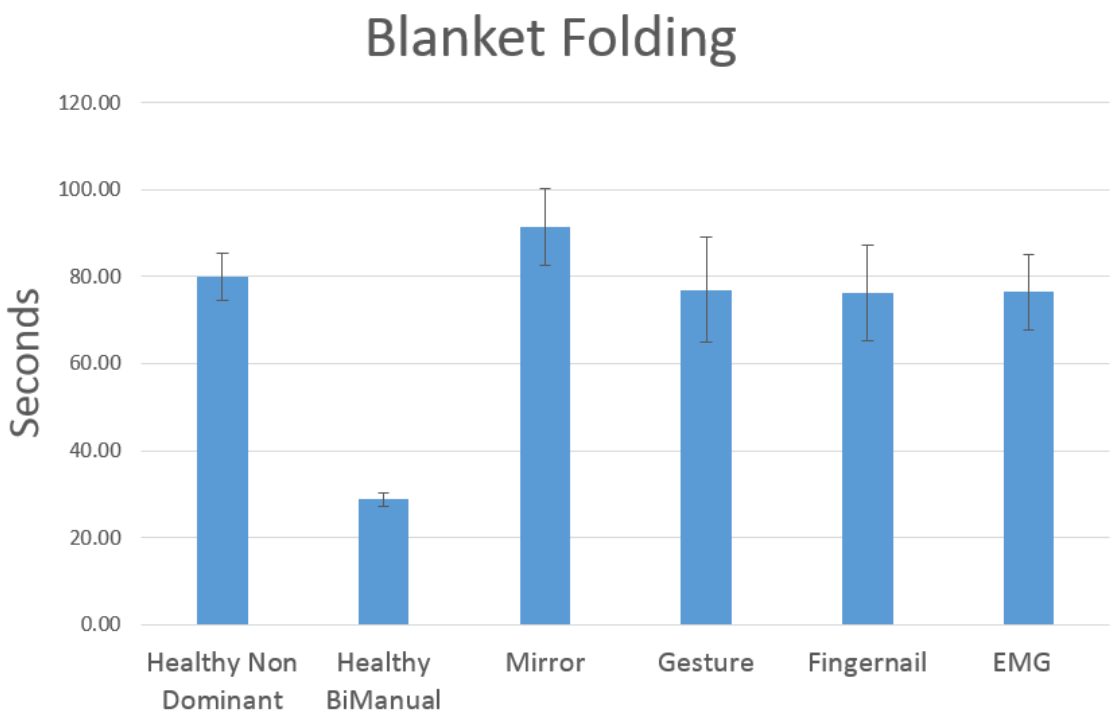

Figure 4.16 Blanket Folding Experiment results for Fingernail Device, EMG, and the other devices discussed earlier.

4.5.4 Tray Carrying Test Results

The volunteers were able to perform the Tray Carrying Test in an average of 25.67 ± 5.24 seconds. The amputee volunteer was not able to perform this test. The comparison of the Tray Carrying Test results with the Fingernail Device, healthy hand, and other devices is depicted in Fig. 4.17.

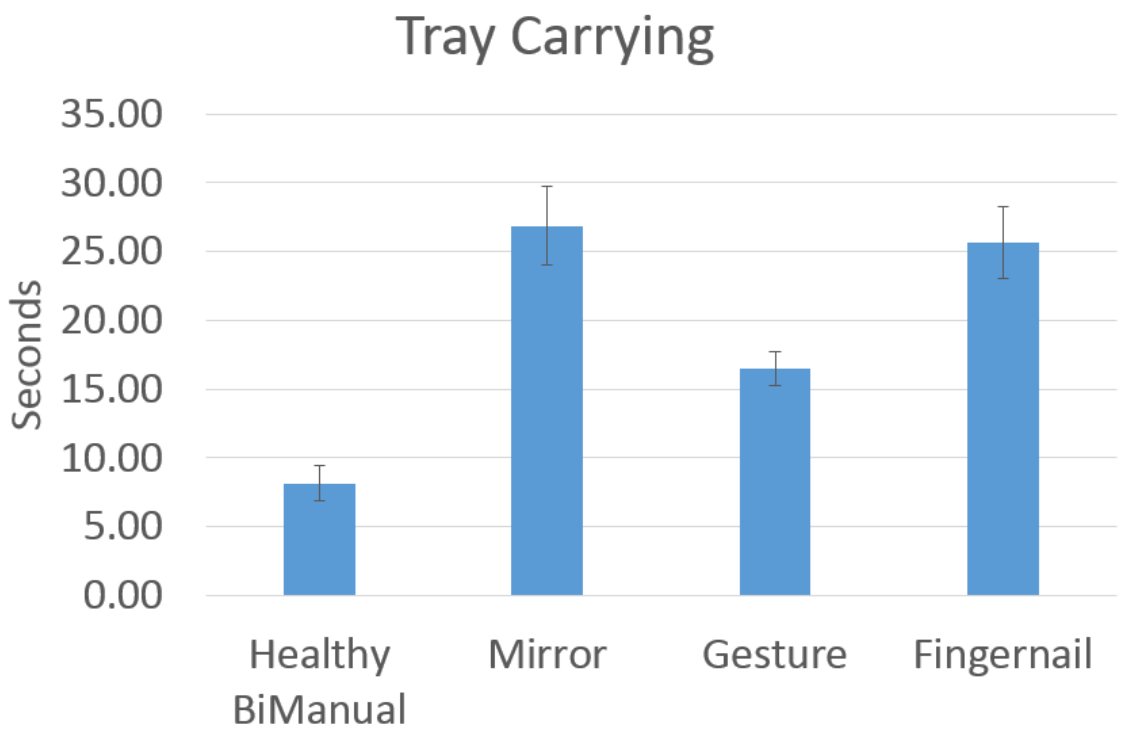

Figure 4.17 Tray Carrying Experiment results for Fingernail Device, healthy hand, and the other devices discussed earlier.

4.5.5 Tissue Characterization Results

The Beer-Lambert law (Equation 2.1) was used to approximate the attenuation of light in tissue. As discussed earlier, human tissue is not a homogeneous medium and the actual scattering and attenuation behavior can't be fully captured by this equation [56] [57]. Nonetheless, this equation provides a good idea of the design limitation imposed by how much light can be detected through two fingers.

Tissue Characterization resulted in slightly different values of the attenuation coefficient μ for one finger and for the system formed by the index finger and the thumb. While the system relies on the relatively easier passage of light from one finger to the other during the performance of the pinch gesture, some loss of light at

the finger-finger boundary was expected. The nature of the attenuation coefficient leads to an important design consideration for the system. There is 27 times less light available at the thumbnail if the light has passed through both fingers as opposed to only one. It is, therefore, easier to implement a thumbnail gesture in which the user wears the fingernail mounted LEDs and performs gestures by touching the pads of any one of the four fingers to the thumbnail (Fig. 4.18, bottom left).

The attenuation coefficient μ was calculated for both the index finger and the thumb. The parameter μ_1 represents the attenuation coefficient through the index finger, μ_2 the attenuation coefficient of the index finger-thumb system, and $\mu_{average}$ the average of both. The parameters were found to be 3.27, 3.46, and 3.37 cm^{-1} respectively. The optical extinction coefficients of various human tissues are well characterized [59] and are not expected to vary for a given type of tissue. Light going from one finger to another may, however, undergo additional losses not necessarily described by the Beer-Lambert law. Factors which influence the performance of the system are the skin color and the thickness of the finger. Skin conditions such as sweat are not expected to affect the performance of the system, as they do not appreciably change optical properties of the underlying tissue. For the design of this system, the important factor is that the light be detectable at the thumbnail when injected from the top of another fingernail when the two fingers are in contact, as illustrated in Fig. 4.7.

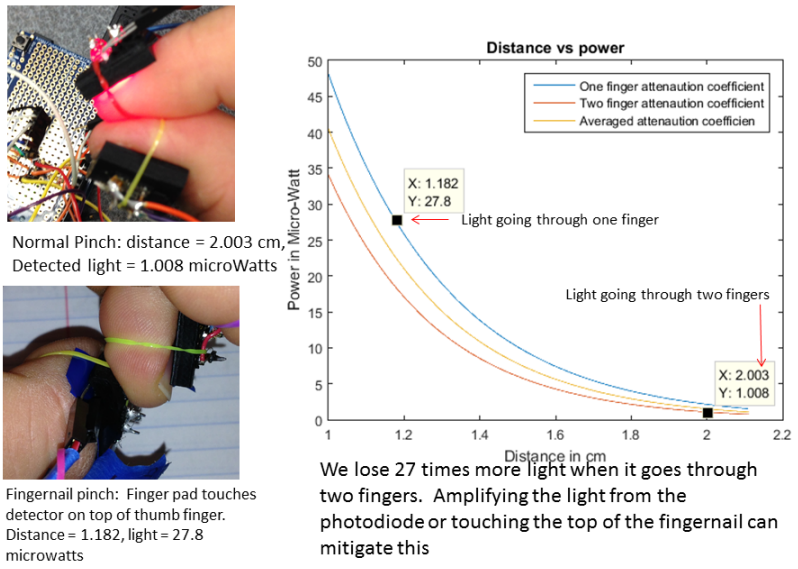

Figure 4.18 Tissue Characterization Experiment Results.

4.5.6 Gesture Performance Experiment

The data from the gesture performance experiment were analyzed by a MATLAB script extract_response_delay (Appendix C.2) in order to characterize the time between the visual prompt and response detected by the system. The system compares the idealized responses from the system based on the prompt given to the user (0, 1, or 2, corresponding to no gesture, gesture 1, or gesture 2), and the response detected by the system (0, 1, or 2). The details of this process are illustrated in Fig. 4.19, where the ground truth data was recorded along with the detection data during the course of the performance experiment. A state of 0 indicates that the user was instructed to perform no gesture. A state of 1 indicates that the LED prompting for gesture 1 was on. A state of 2 indicates that the LED prompting for gesture 2 was on. The graph in the upper right hand corner overlays both the prompted and the

detected gesture signal on the same graph for the entire experiment. A MATLAB script called extract_response_delay was created to automatically detect events on the rising edge of either signal, while ignoring brief transient detection errors.

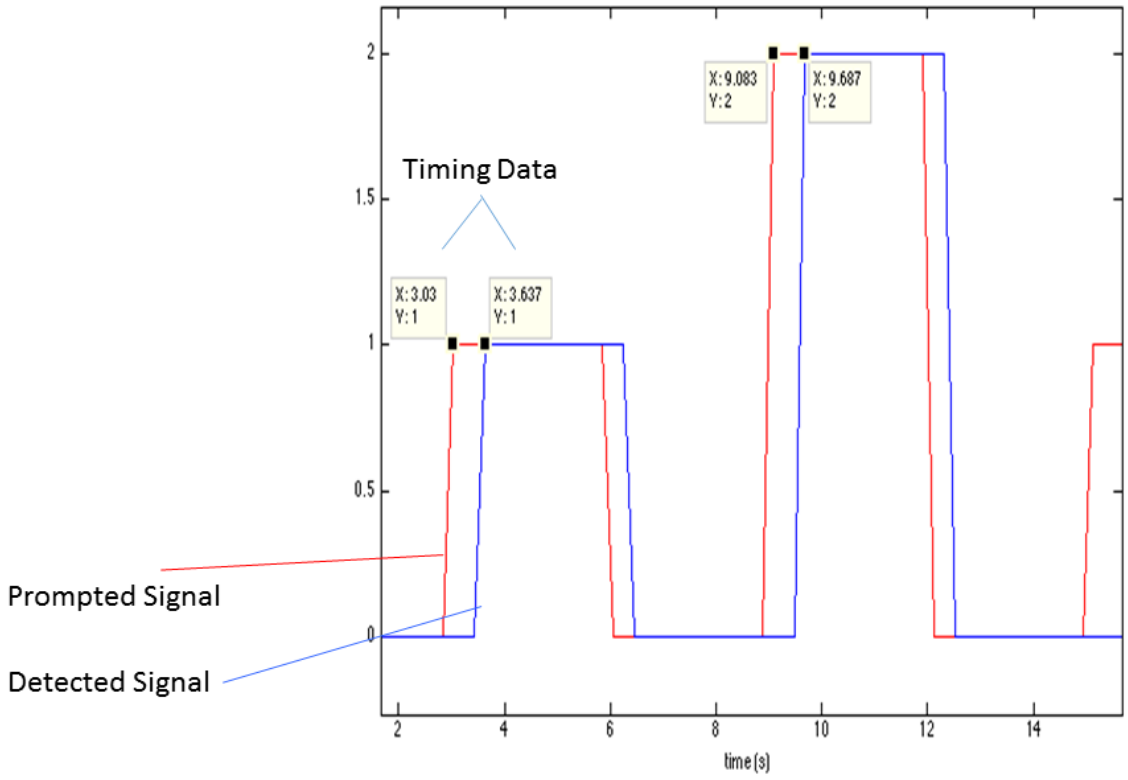

Figure 4.19 Performance Experiment Details. The red trace indicates the prompt given by the LED prompter illustrated above. The blue trace indicates the gesture as detected by the gesture detection system.

The MATLAB script extract_response_delay identified touch events based on the rising edge of the detected gesture signal, while ignoring transient misclassifications such as the one identified as Detection Noise in the figure. The program calculated the time difference between the touch prompt and the detection of the touch event to be 0.681 ± 0.15 seconds (mean delay). This is a combination of the

gesture performance delay inherent in human perception and the gesture detection delay. A total 8 events performed by the volunteer in response to a visual prompt were correctly identified (Fig. 4.20). While the start of each event was identified correctly, some noise can be seen in the last three events.

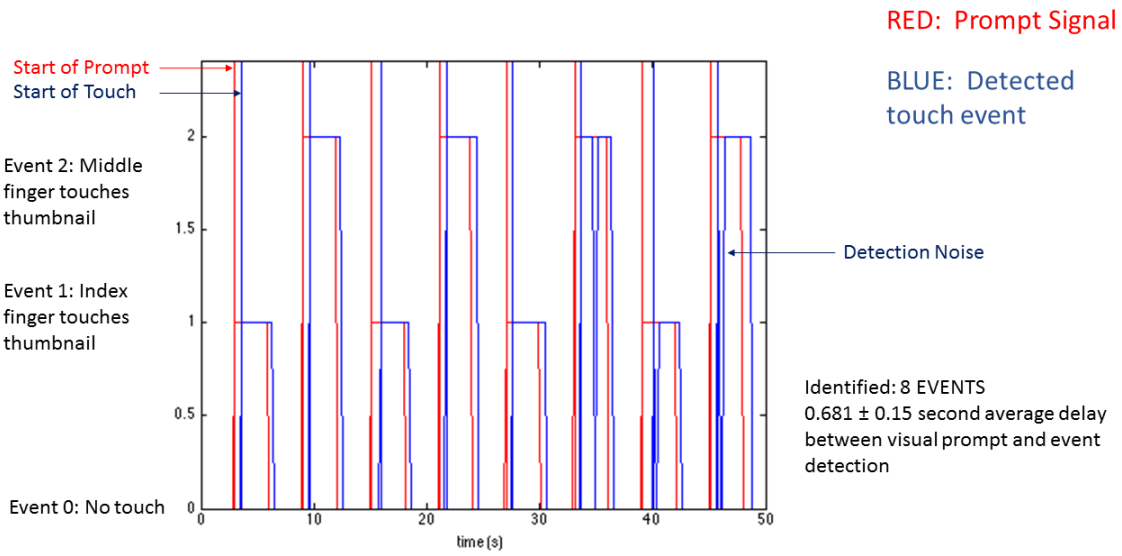

Figure 4.20 Gesture Performance Experiment Results.

4.5.7 FFT Based Detection Through Two Fingers

All 10 gestures performed by the subject were correctly identified by the FFT based method, as can be seen in Fig. 4.21.

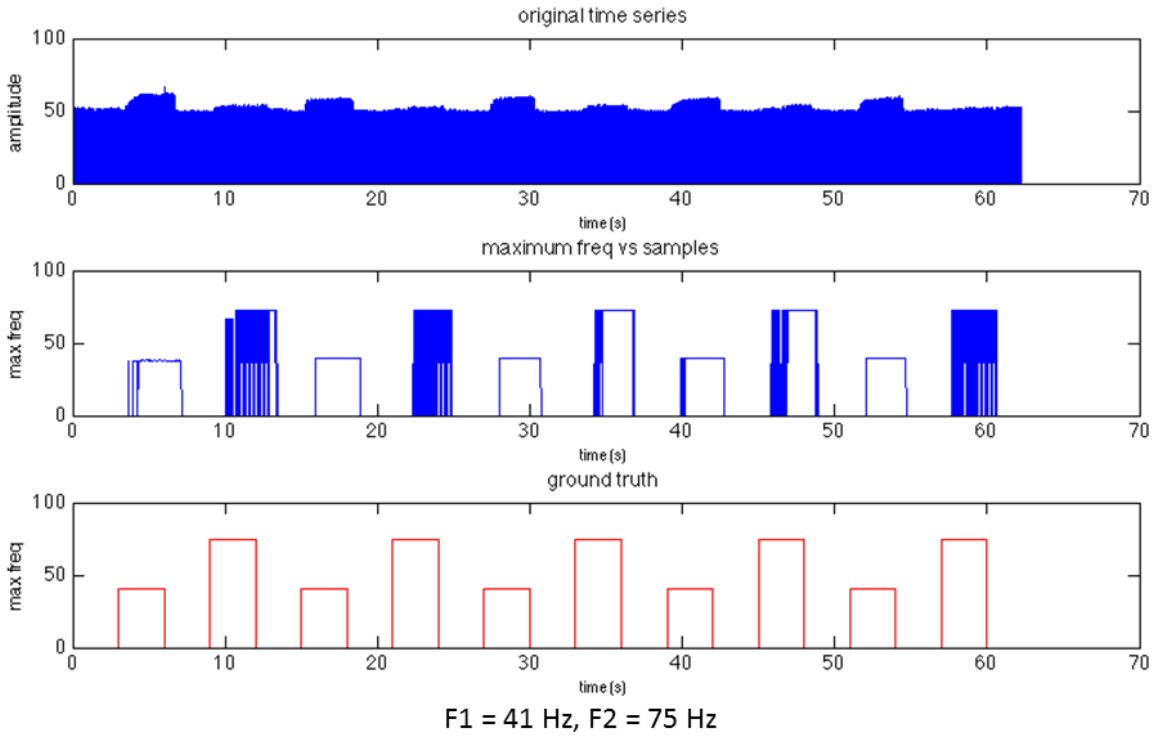

Figure 4.21 FFT Experiment Results.

4.6 Discussion

4.6.1 Gesture Performance Experiment

The gesture performance experiment indicates that the response detection time was around 0.6 seconds. While this is outside the accepted range of human stimulus to visual stimuli (typically around 0.25s), several factors must be kept in mind: first, the device and the detection algorithms are still in their preliminary stages and streaming data to a PC from the device induces a significant amount of delay to the microcontroller. Second, this experiment was performed on the non-dominant hand, using a "thumbnail gesture" in which the user touches the top of the thumbnail with one of the finger pads. This is a somewhat unnatural pose and may not facilitate the fastest device performance.

4.6.2 FFT Based Detection through Two Fingers

The FFT based detection method was the only method which could reliably identify and distinguish between signals from LEDs mounted on different fingernails through light transmitted through both fingers, with the hardware used. It also lends itself to a very low cost implementation of the system in which the fingernail mounted devices only contain simply circuitry and a battery to flash at a characteristic frequency, with all of the system logic residing in a thumbnail mounted device.

To select the optimal frequency pair a frequency pair search was conducted which yielded many similarly optimal results (140, 100), (60,40), etc. The frequency pair chosen for the actual experiment was (41,75). The LED on one finger was flashed at 41 Hz while the LED on the other finger flashed at 75 Hz. The light coming through both fingers was detected by the photodiode facing the thumb. This was due to the fact that flashing the LEDs at high rates on the Arduino microcontroller board tended to introduce errors.

4.6.3 Results of Single Digit Response Time

The results of the single digit response time experiment are presented in Table 4.2.

Table 4.2 Average Single Digit Response Times in Seconds. The experiment was performed on two male and two female right handed volunteers ages 22-47 with no manual dexterity problems.

	index	middle	ring	pinkie
L Thumbpad	0.32	0.38	0.45	0.52
R Thumbpad	0.36	0.39	0.48	0.48
L Thumbnail	0.39	0.37	0.38	0.44
R Thumbnail	0.44	0.42	0.40	0.44

Even though the subjects were all right-handed, the fastest response times appear to be on the left hand. Fig. 4.22 summarizes the results grouped by finger. In Fig. 4.23, the aggregate response times sorted from fastest to slowest can be compared. While individuals vary in which gesture they perform the fastest for the more difficult gestures (such as touching either the thumbpad or the thumbnail with the pinkie or the ring fingers), each volunteer consistently performed left thumb pad to index finger gesture the fastest, followed by "right thumb pad to index", "left thumbnail to middle," and "left thumb pad to middle." In fact, thumbpad to index gesture was significantly faster on the left hand ($0.316 \pm 0.121s$) than on the right hand ($0.361 \pm 0.13s$) ($p < 0.005$).

Gestures being faster on the left hand is surprising given that all the volunteers were right-hand dominant. One confounding factor may be that every volunteer was also trained in touch typing using an English QWERTY style keyboard, in which the left hand keys are used more frequently. Even though it is unlikely to affect the final design of the Fingernail Device, this test should be repeated with left-handed subjects and non-typists.

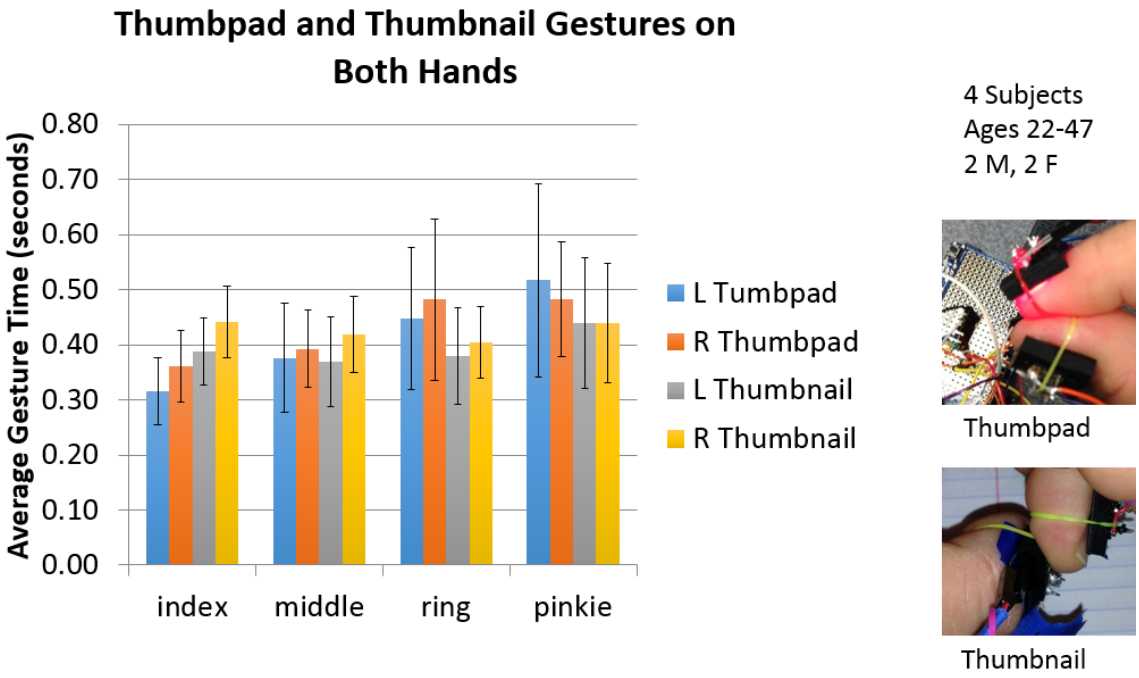

Figure 4.22 Single digit response time.

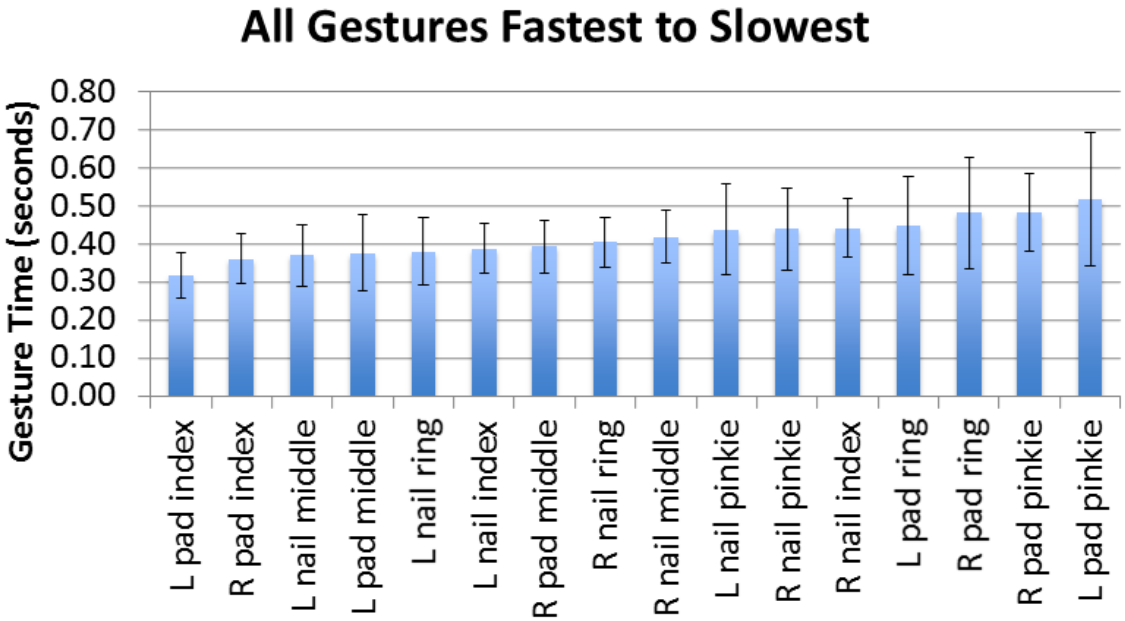

Figure 4.23 Single digit response time, sorted from fastest to slowest. Each bar represents 120 trials.

In clinical practice, it may be useful to perform a timing test similar to this one on the potential prosthetic user, and map the most frequently used prosthetic device functions to the gestures which can be performed fastest by that individual. It should also be kept in mind that this experiment used only healthy volunteers with no history of arthritis other disorders. Two volunteers (not part of the results) were rejected due to their inability to comfortably perform all possible gestures.

4.7 Conclusion and Future Work

Even without any further improvement, we believe the device to be sufficiently advanced for our intended purpose, which is the control of prosthetic devices. Four gestures can very easily be mapped onto four prosthetic device configurations such

as "open" and three grasp positions such as lateral grasp or pinch grasp. Many more gestures are easily available including, but not limited to, chord gestures in which multiple fingers can touch the thumb, gestures in which touching the pad of the thumb has a different meaning than touching the thumbnail. In fact, using a variant of the system discussed here in which LEDs and photodiodes are present on both the fingernail facing side and the outside facing side of the device, readily available gestures go up to 12 from four (e.g., finger pad touches thumb pad, finger pad touches thumb nail, thumb pad touches finger nail).

Time based gestures (such as hold, long hold, etc.) are very common in EMG based prosthetic control and can be adapted to our system. In addition, concepts can be borrowed from PC interfacing such as double-click and triple-click. Using double tap and triple tap would also increase the value of the system by making more gestures readily available.

4.7.1 Future Work

4.7.1.1 Multiple Digit Response Time Experiment

The purpose of this experiment will be to determine the response speed of the average subject on one of four randomly prompted gestures, and it will be performed using the Fingernail Device itself in order to characterize the response time with the device. This test is similar to the Single Digit Response Time Experiment described above but involves a choice on the part of the user. A well-known equation now known as Hick's Law [97] states that reaction time increases in proportion to the base 2 logarithm of the number of choices available.

The subjects will view an array of 4 LEDs, randomly prompting the subject to perform one of 4 thumb pad or thumbnail gestures depending on the experiment. The prompt time will be randomly varied between 0.5 to 2.0 seconds. Each subject will perform 10 gestures in each of 4 configurations (left hand thumb pad, left hand thumbnail, right hand thumb pad, right hand thumbnail) before moving on to the next configuration. Each subject will repeat the entire set of experiments three times. In order to determine gesture performance and accuracy, the following parameters will be recorded: prompted gesture, performed gesture, reaction time.

4.7.1.2 Untethering and New Hardware Implementation

Future work includes implementing the FFT based system in hardware, untethering the fingernail mounted LED emitters from the Fingernail Device, adding wireless capability to the thumb based receiver, using lower powered LEDs in the infrared range in order to not distract the user, and amplifying the signal from the photodiode in order to reduce LED size and power (a proof-of-concept device using an Arduino XBee shield to transmit detected events has been demonstrated). Any future system will also have to include the ability to turn the system on and off – a gesture such as double-tap or triple-tap may be appropriate for such a use, as it is unlikely to be accidentally triggered. The final embodiment of the device and finger attachment method will also have to be decided upon.

It is also possible to implement a system that is similar in concept to the one discussed here by using methods other than light detection. Such systems might include placing RFID stickers with different identifiers on the fingernails and embedding an RFID reader on the thumbnail, or creating passively powered inductor-capacitor

based circuits embedded in fake fingernails whose presence near the thumb can be detected by specialized circuitry (similar to metal detectors).

We believe our system can increase the quality of life of amputees by providing a ubiquitous and always on interface to their prosthetic devices while staying out of their way when not needed. The same reasons may also make this device useful outside the amputee community.

CHAPTER 5

USING BILATERAL MOVEMENTS TO TRAIN EMG AND FMG CLASSIFIERS

5.1 Problem Statement

The major goal of prosthetic device control interfaces is to provide the user with intuitive control, in which the user simply uses the prosthetic device as it were a natural appendage. In transradial amputees (amputees who have a limb missing below the elbow), there are usually remaining muscle groups which move in accordance with the signals from the brain, even though the hand and digits may not be present. Existence of these residual muscles provides a readily available source of information on user intent, and presents the possibility of enabling multiple degree of freedom (DOF) prosthetic device control. Earlier chapters in this work have explored systems which can be used when these signals are not usable.

This chapter discusses the development of a field-deployable system for natural robotic prosthetic device control which combines readily available electromyography (EMG) and force myography (FMG) signals from the residual limb of the amputee. In this system, training of the classifier can be performed as needed by the amputee utilizing a tracked glove on the intact hand and an EMG/FMG sensor array on the residual limb. The system will prompt the user to perform bilateral mirrored hand movements, and use the kinematic information from the intact hand and the muscle electrical activity (EMG) and surface muscle movements (FMG) from the residual limb to adjust the classification system.

5.2 Experiments Performed

5.2.1 Data Acquisition

A custom EMG sensor was designed and built using an AD623 instrument amplifier and three NTE948 operational amplifiers depicted in Fig. 5.1. The output of the EMG circuit was sampled with an Arduino UNO Microcontroller board and streamed to a laptop computer at a speed of 115,200 bits per second. Eight of the EMG circuits were created on an Arduino Mega shield for use with the Arduino Mega Microcontroller Board.

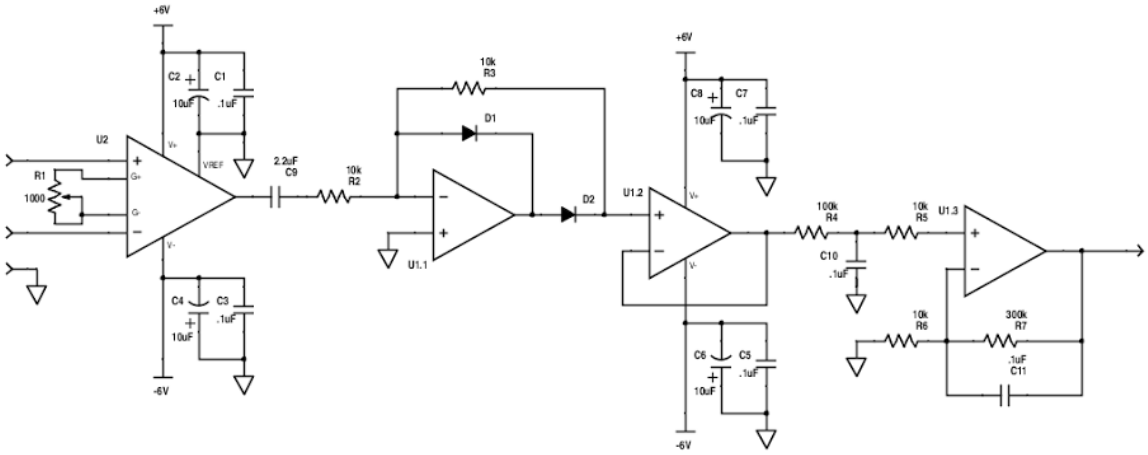

Figure 5.1 Custom EMG Circuit Used to Acquire the Data. The first stage on the left is the instrument amplifier (AD623). The amplifier was designed and built by Mr. Chad Bonner and Mr. William Kitchen. Diagram used with permission.

Experiment 1

At the time of this experiment, only a single EMG unit had been built and validated. In order to simulate the data from multiple sensors, the subject was asked to perform the same task multiple times while the electrodes were placed in multiple

parts of the arm. This experiment demonstrates the validity of this task by acquiring data at separate times from the back of the arm and forearm. The correspondence between different time points was established by precisely timing the prompts given to the user.

Electrodes were placed on both the anterior and posterior sides of the forearm. For each run of the experiment, the subject was prompted by a blinking LED to flex and extend the forearm. The prompt (1 for flex, 0 for extend) was logged along with the EMG data. The data for this experiment are shown in Fig. 5.2.

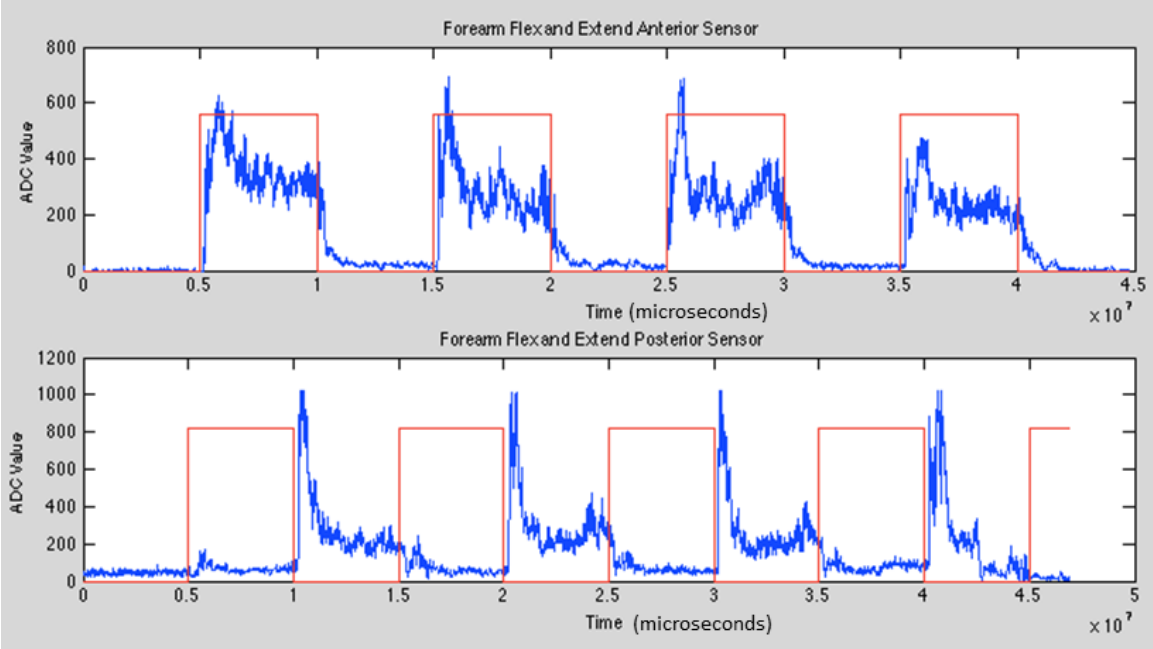

Figure 5.2 Data acquired from sensors on the anterior and posterior of the forearm during the same movement.

In Fig. 5.2, the red squares correspond to the prompt given to the subject to flex the forearm (this signal has been scaled to 80% of the maximum value of the

EMG signal for visibility).

The data from Fig. 5.2 were used with the MATLAB neural network training functionality to train a neural network (a "patternnet") with two inputs, 10 hidden layers, and one output. Of the three datasets acquired, two of them (Dataset 2 and Dataset 3, shown on the figure) were utilized. One of the datasets (Dataset 3) was used for training, while another (Dataset 2) was used for validation. The prompt given to the user to flex or extend the forearm was used as the training output. The output of this experiment is depicted in Fig. 5.3. The vertical axis in the bottom of Fig. 5.3 represents the predicted action: 1 for flex, 0 for extend.

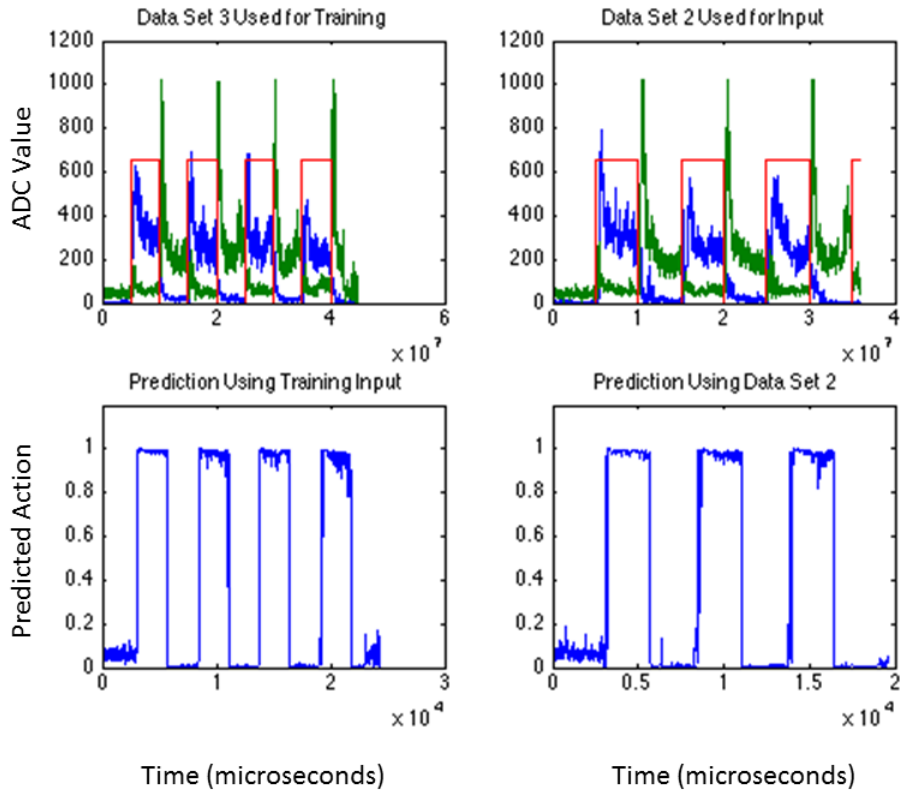

Figure 5.3 Training set for flexion and extension and predicted output using both the training set (left bottom) and the validation set (right bottom).

Experiment 2

This experiment is an attempt at data collection for distinguishing between index finger and pinkie flexion. The correct locations for index finger flexion (flexor digitorum superficialis) and pinkie flexion (flexor carpi ulnaris) were identified on the subject from and verified by palpating the locations while the subject performed the index and pinkie flexion tasks. The experimental setup is depicted in Fig. 5.4.

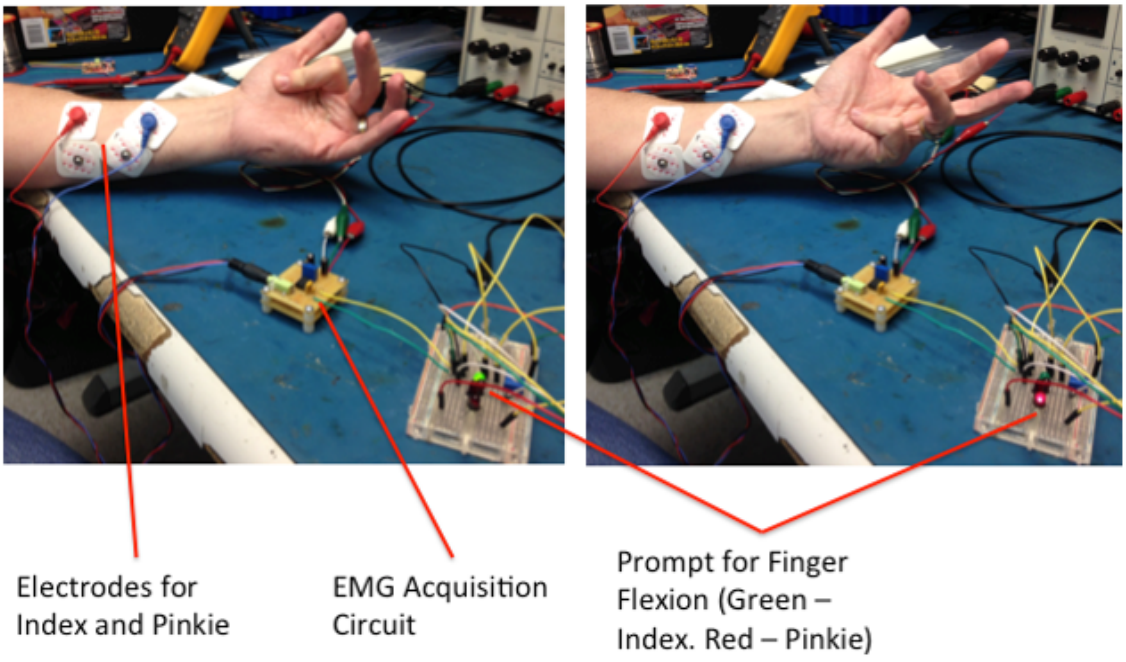

Figure 5.4 Experimental setup to acquire data to distinguish between pinkie and index finger flexion.

The experiment was run in multiple iterations by changing the electrode placement and asking the subject to perform the same task. The LED seen on the breadboard prompts the user to flex either the index finger or the pinkie, and the action prompted by the LED (in this case, index finger or pinkie flexion) is used as ground truth for neural network classification.

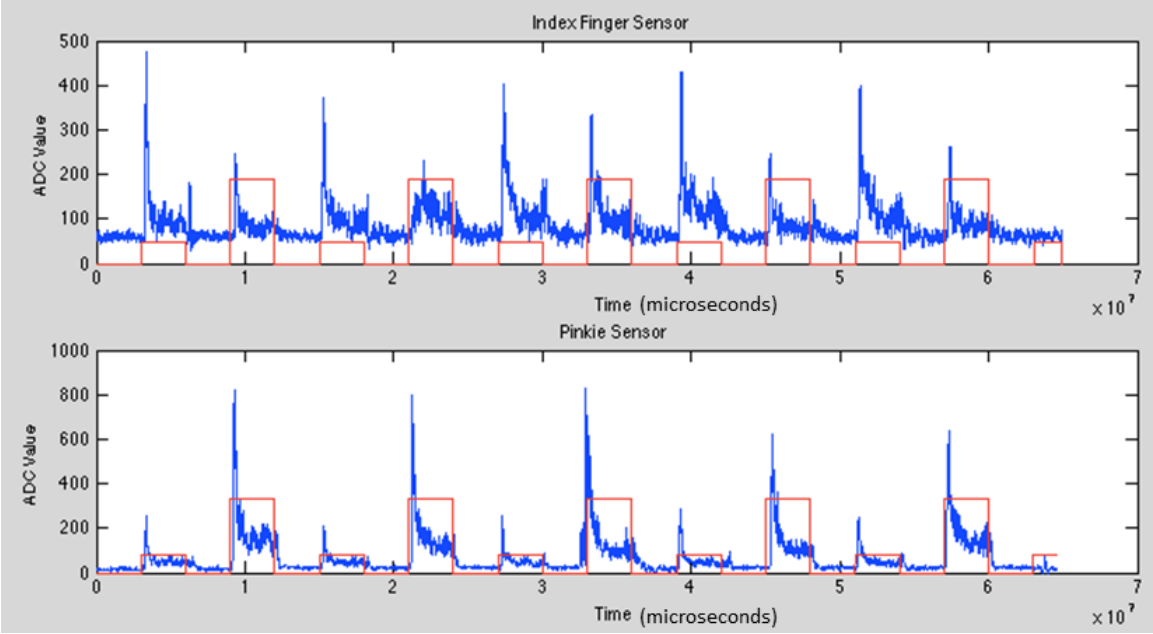

Figure 5.5 Preliminary data from the two finger discrimination experiment.

In Fig. 5.5, the top graph shows the EMG readout from the flexor digitorum superficialis muscle (used for index finger flexion) and the bottom graph shows the EMG readout from the flexor carpi ulnaris muscle (used for pinkie flexion). The shorter red lines indicate the prompt for index finger flexion, and the taller red lines indicate the prompt for pinkie flexion. The subject was instructed to rest when there was no prompt.

The data from Fig. 5.5 were used to train a neural network with two inputs, 10 hidden layers, and one output. The prompt given to the user to flex the index finger or the pinkie was used as ground truth for training input. The output of this experiment is depicted in Fig. 5.6.

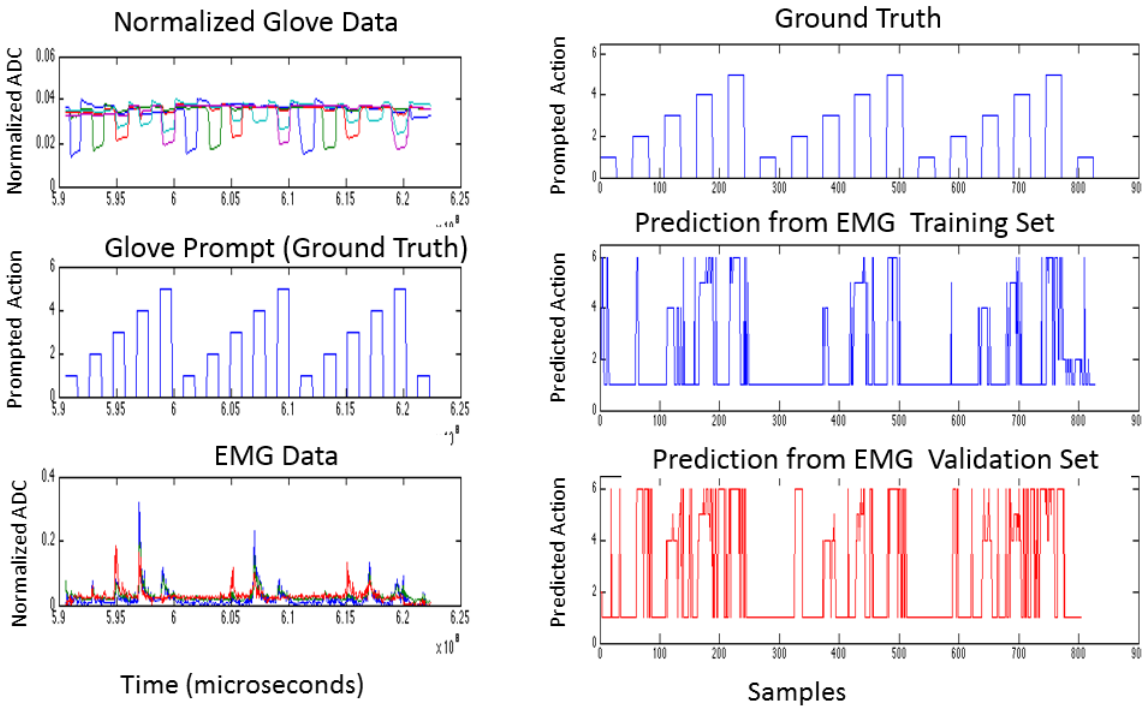

Figure 5.6 Output of neural network for discrimination between index and pinkie fingers.

The vertical axis in the bottom of Fig. 5.6 represents the predicted action: 3 for pinkie, 2 for index, 1 for no action. As can be seen in the bottom right of the image, the system was able to predict middle finger, index finger, and pinkie movemens, albeit with some noise. Performance in future experiments can be improved by having a larger number of electrodes.

5.2.2 Multichannel EMG recordings on Healthy Subjects

A multichannel version of the EMG sensor depicted in 5.1 was created on a custom "shield" for the Arduino Mega microcontroller board as depicted in Fig. 5.7.

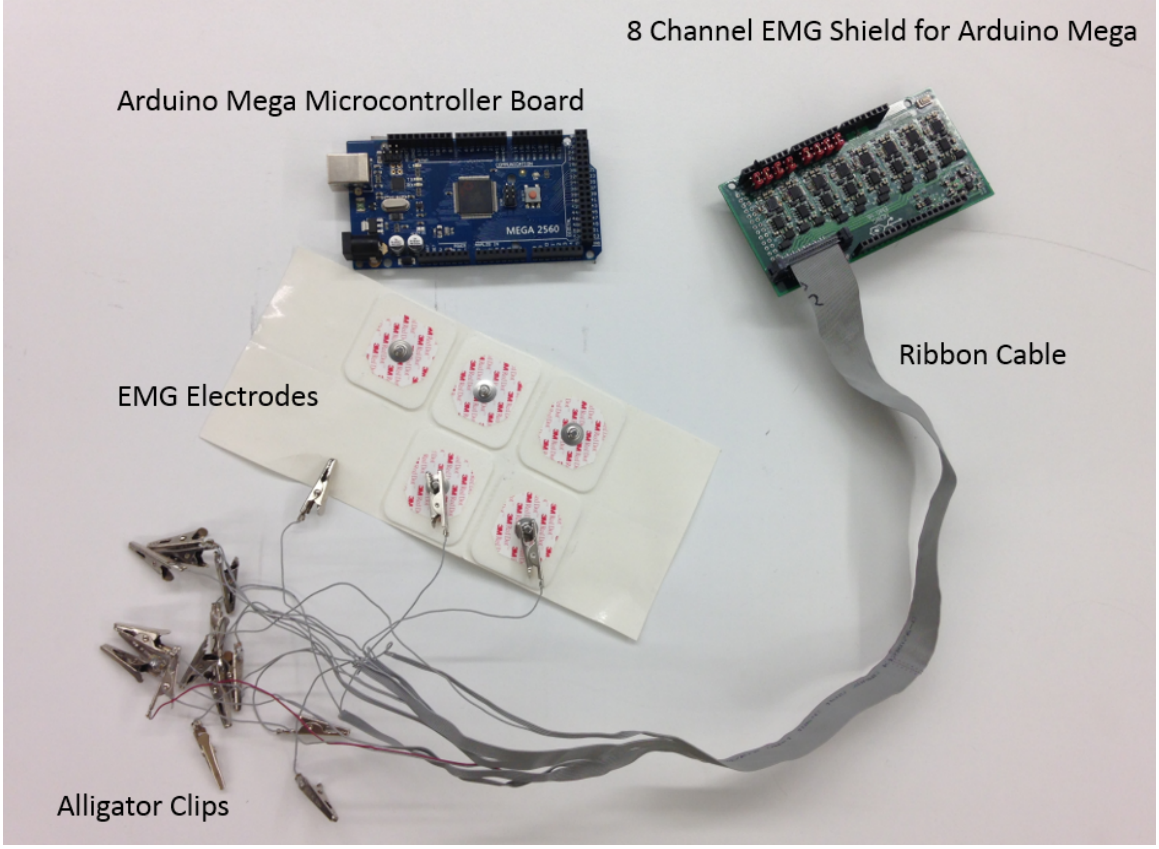

Figure 5.7 Custom EMG Shield for Arduino Mega Microcontroller Board.

The board was affixed to the subject's wrist by means of affixing a square of Velcro underneath it and having the subject wear a bracelet made from Velcro as depicted in Fig. 5.8. Finger flexion data were acquired from a custom built data glove with piezoresistive flexion sensors affixed to each finger. The LED prompter instructs the subject to flex or open a specific finger. The prompter values are logged as ground truth along with the timing information, flexion of each finger, and the EMG data.

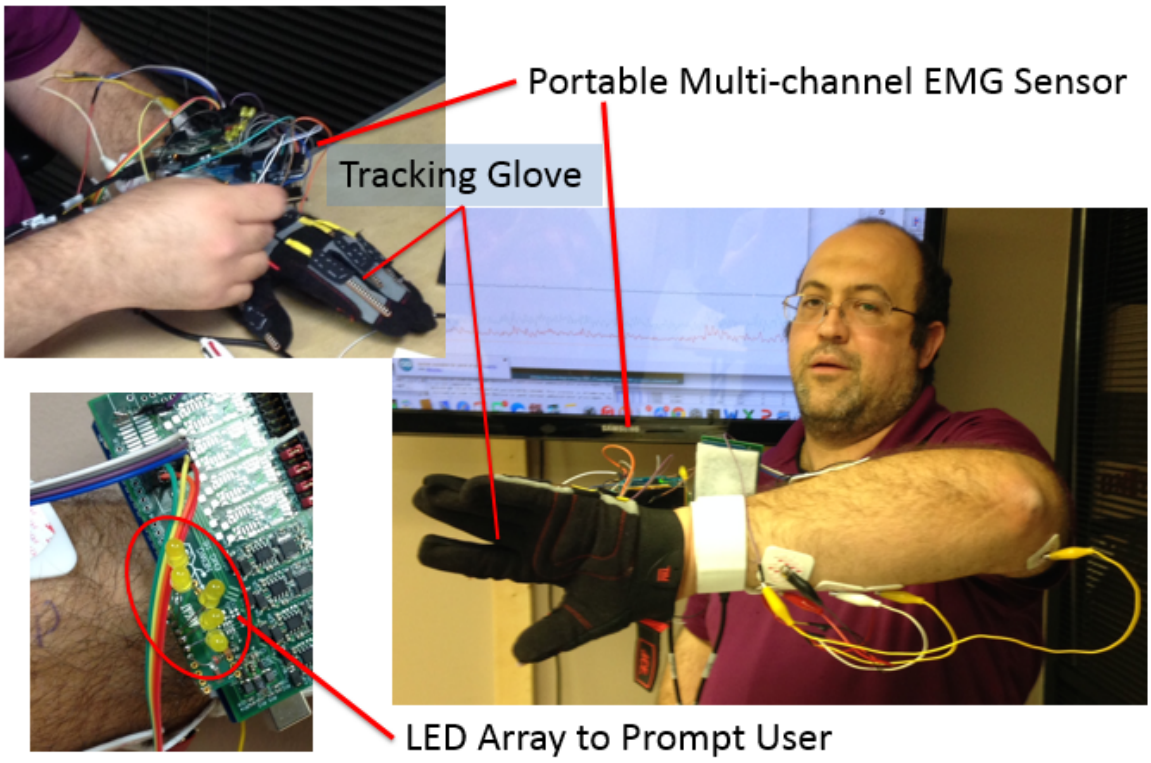

Figure 5.8 Recording setup for healthy subject.

In order to be useful to the classification algorithm, the glove data need to be normalized and processed to represent the flexion of a single finger. While it may be useful for future incarnations of the system to predict the degree of flexion of each finger, the current system assumes that the user intends to either close or open a single digit.

The neural network based training scheme to be used in this system is comprised of two parts: The Glove Preprocessor Neural Network (GPNN) and the User Intent Predicting Neural Network (UIPNN). The GPNN)= transforms the glove data into discretized signals representing user finger flexion. These signals are used as ground

truth while training the UIPNN. The function of the UIPNN is to turn EMG and FMG data acquired from the subject into robotic prosthetic device movements in real time. The training scheme is illustrated in Fig. 5.9.

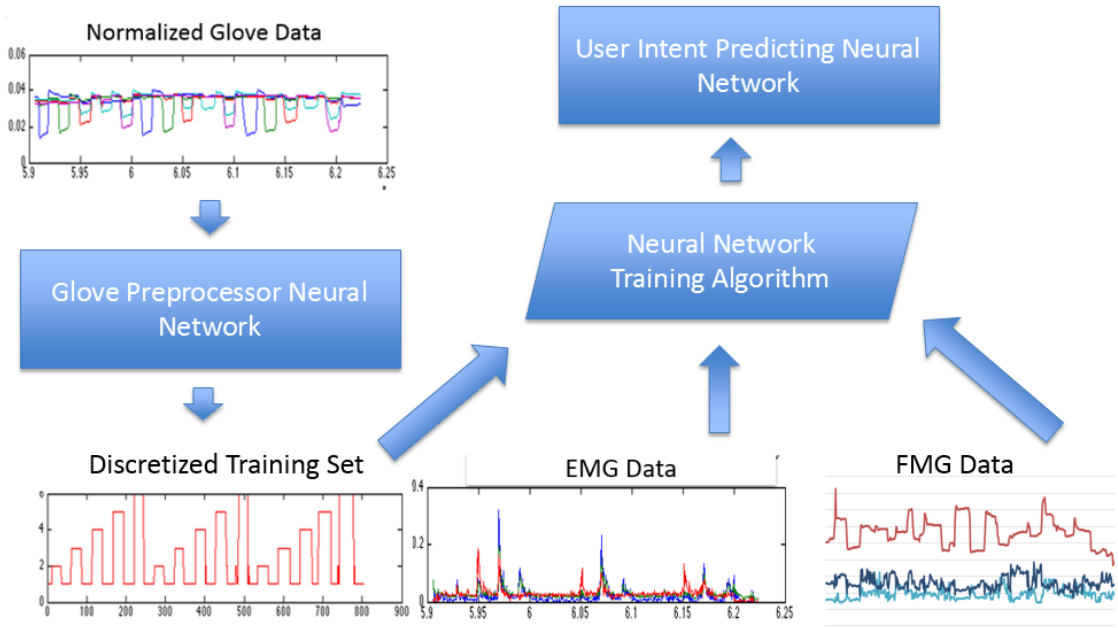

Figure 5.9 Overall Neural Network training scheme.

The execution of the system, once trained, is depicted in Fig. 5.10.

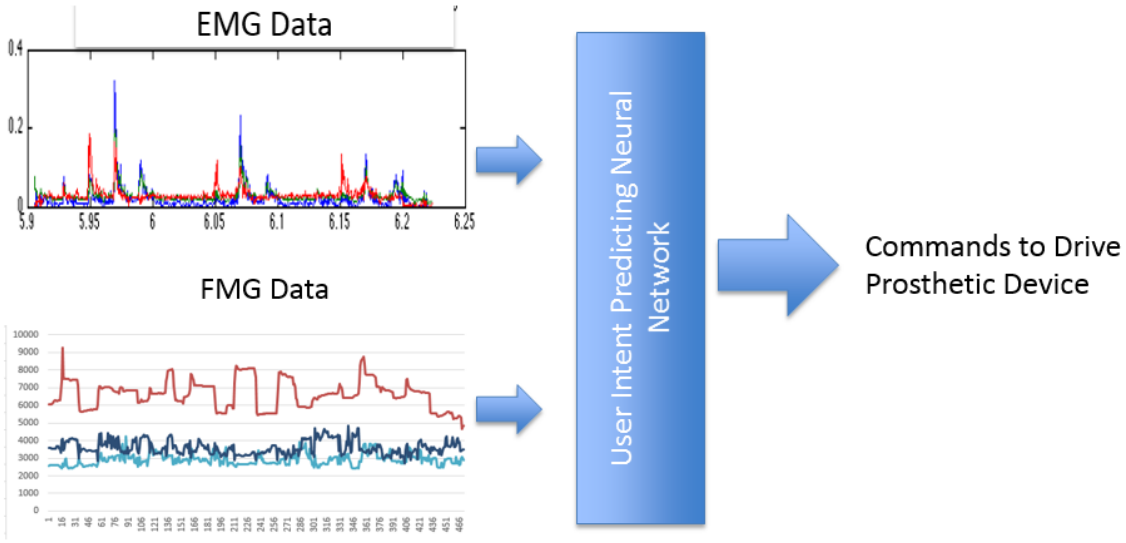

Figure 5.10 Operation of the Neural Network once trained.

The GPNN (implemented using MATLAB's built in patternnet) with 30 hidden neurons, 5 inputs, and 6 outputs was trained on the LED prompter data as ground truth. This network takes the flexion of each finger and outputs one of 6 classes: No Action; Thumb Flexion; Index Finger Flexion; Middle Finger Flexion; Ring Finger Flexion and Pinkie Flexion.

The trained GPNN was able to classify this dataset successfully, as seen in 5.11. The GPNN was trained using the LED Prompter data as ground truth and the glove flexion data as input. The purpose of this neural network is to act as a preprocessor to the acquired glove data in order to feed it into the training algorithm for the UIPNN. Once trained, the GPNN was able to correctly predict the movement of each digit from the data representing the flexion of the piezoresistive sensors on each digit of the glove.

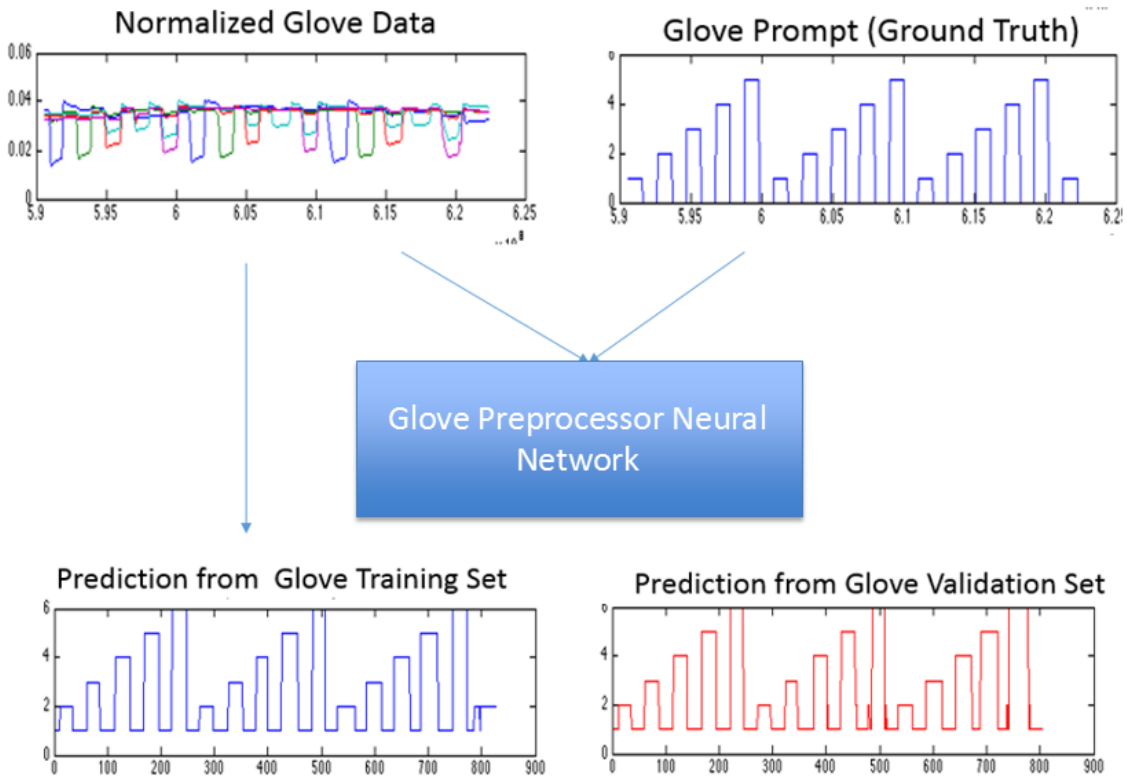

Figure 5.11 The Glove Preprocessor Neural Network (GPNN) trained on LED prompter data.

The ground truth data from the prompter LEDs (Fig. 5.12) were used to train a neural network on the acquired EMG data in order to predict digit movement as seen in Fig. 5.13. This device typically instructs the user to flex one or more fingers, or relax the hand (when no LEDs are on). The program running on the microcontroller records the action of these LEDs as ground truth to be used for neural network training.

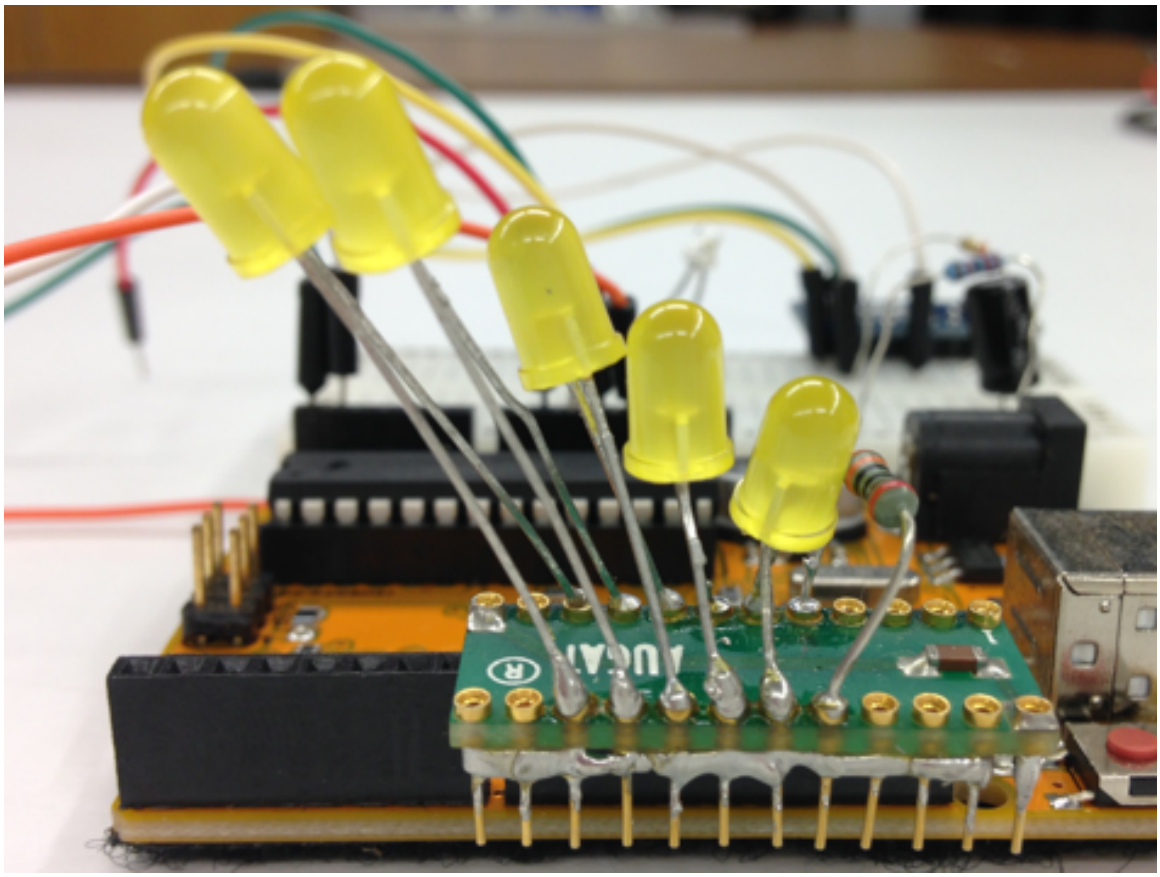

Figure 5.12 LED Prompter Device.

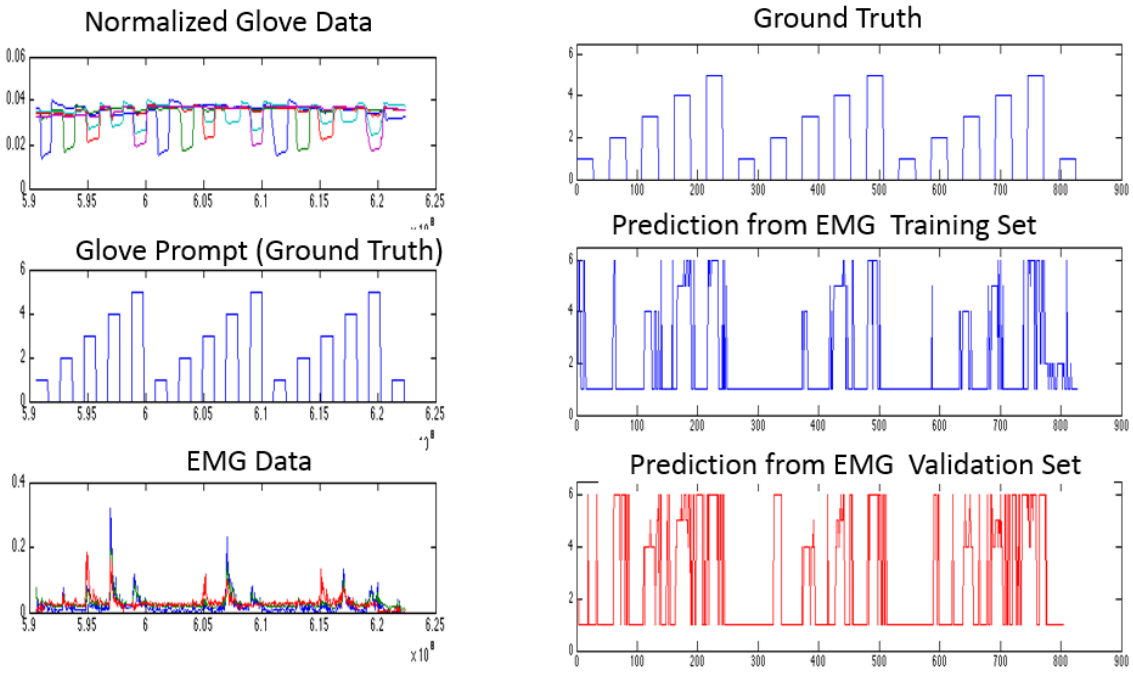

Figure 5.13 Preliminary digit position prediction from EMG data.

Fist Clenching Experiment

The healthy volunteer was instructed to clench and relax his fist as prompted by a single LED. The dataset captured from the experiment is depicted in Fig. 5.14.

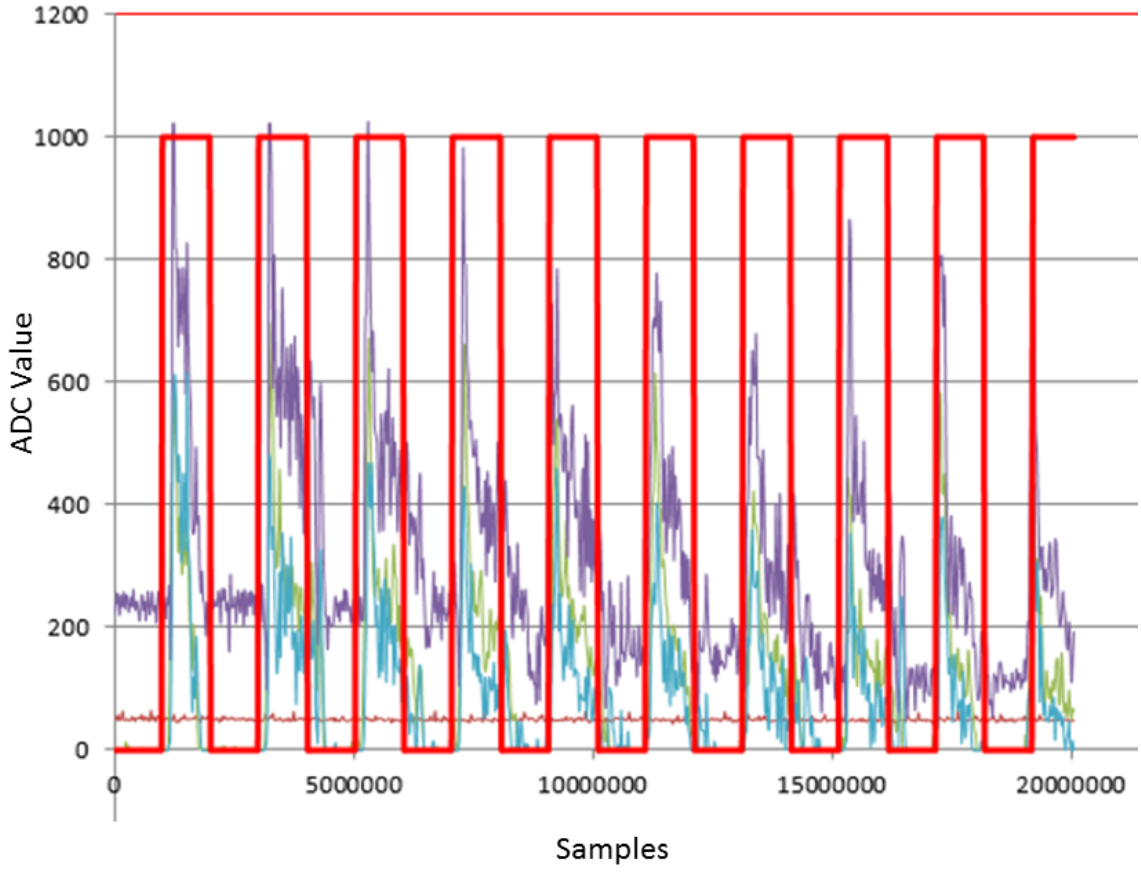

Figure 5.14 EMG Data from Fist Clenching Experiment.

The purpose of this experiment was to validate the hardware. As can be seen in Fig. 5.14, three of the four electrodes respond to the fist being clenched as expected (red lines indicate prompt to clench fist).

5.2.3 EMG Recording on Amputee

Data from a 26 year old male with type 5 symbrachydactyly on his left hand was recorded using a Delsys Bagnolli 16 system with 8 EMG electrodes using the prompter and the glove depicted in Fig. 5.4. The data output from the Arduino

device driving the prompter LEDs (Fig. 5.12) was synchronized with the data acquisition system (DAQ) on the Bagnolli 16 system by wiring the output of the first prompt LED to one of the inputs of the DAQ. The shared pulse was used to align the two datasets.

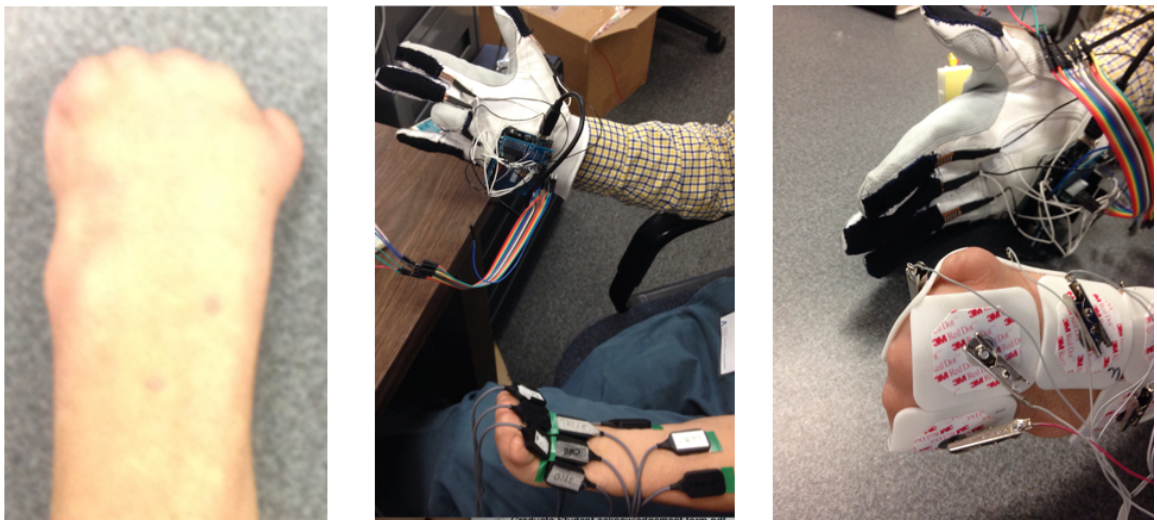

Figure 5.15 EMG Setup on amputee. Left: affected hand with type 5 symbrachydactyly. Middle: Setup with Bagnolli 16 System and Prompter LEDs Right: Setup with Arduino, Tracking Glove, and Prompting LEDs.

The same volunteer was also instructed to perform bilateral movements while wearing the glove and the Arduino based system. The setup is depicted in Fig. 5.15.

The ground truth, glove, and EMG data acquired from the amputee with the Arduino based system is depicted in Fig. 5.16. In this experiment, the action recorded from the LED prompter array (top, Fig. 5.16) is assumed to the be ground truth. The prompted actions are: 0: no action (no LEDs on), 1: flex thumb, 2: flex index finger, 3: flex middle finger, 4: flex ring finger, 5: flex little finger.

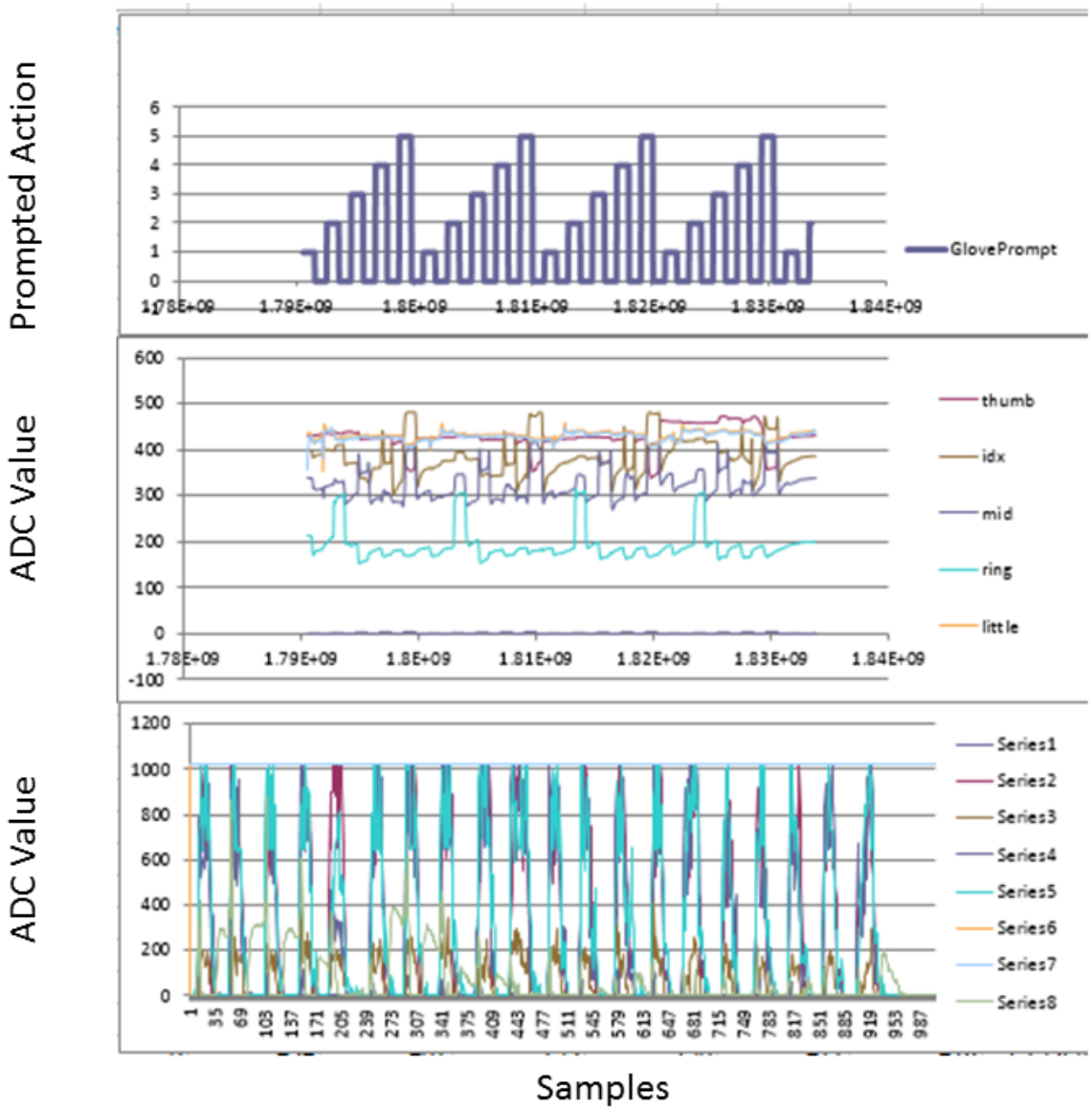

Figure 5.16 Data from amputee. Top: Ground truth based on LED prompts. Middle: Glove Data. Bottom: EMG channels.

As can be seen from the bottom of Fig. 5.16, all the EMG channels appear to respond the same way to all the stimuli, making individual finger flexion identification difficult, if not impossible with the current experimental setup. This may be due to

the fact that the individual who volunteered for the experiment has a congenital hand defect and is possibly unable to control individual muscle groups linked to various fingers. Future work on this particular type of congenital condition can include recording from smaller areas on the knuckles and using tactile feedback to train the subject to generate finger-specific signals.

5.2.4 Force Myography Recording

A 16 channel force myography setup was created from four ADS1115 16 bit analog to digital converters connected to an Arduino Uno device via the I^2C protocol. Eight $FlexiForce^{TM}$ sensors were connected to the device.

One 41 year old right handed male subject was recruited and signed an informed consent form. A mold of the subject's left forearm was created using InstaMorph thermoplastic after heating it in boiling water, flattening it into a thin sheet, and letting it cool to a point where it is comfortable to touch with the bare hand (around 40 degrees Celsius).

Eight locations on the subject's forearm were identified via palpation while the subject was asked to move each digit. The locations chosen were those in which the greatest amount of movement could be felt and corresponded roughly to Fig. 2.12. These locations were marked with a pen and once all locations were identified, the force sensors were affixed with a temporary adhesive (BluTack, Bostik Australia Pty Ltd, Victoria, Australia).

After placing each sensor, the plastic cover was placed on the subject's forearm and the subject was instructed to perform finger movements while the ground

truth from the LED Prompter and the force sensor values were recorded. The setup described here can be seen in Fig. 5.17.

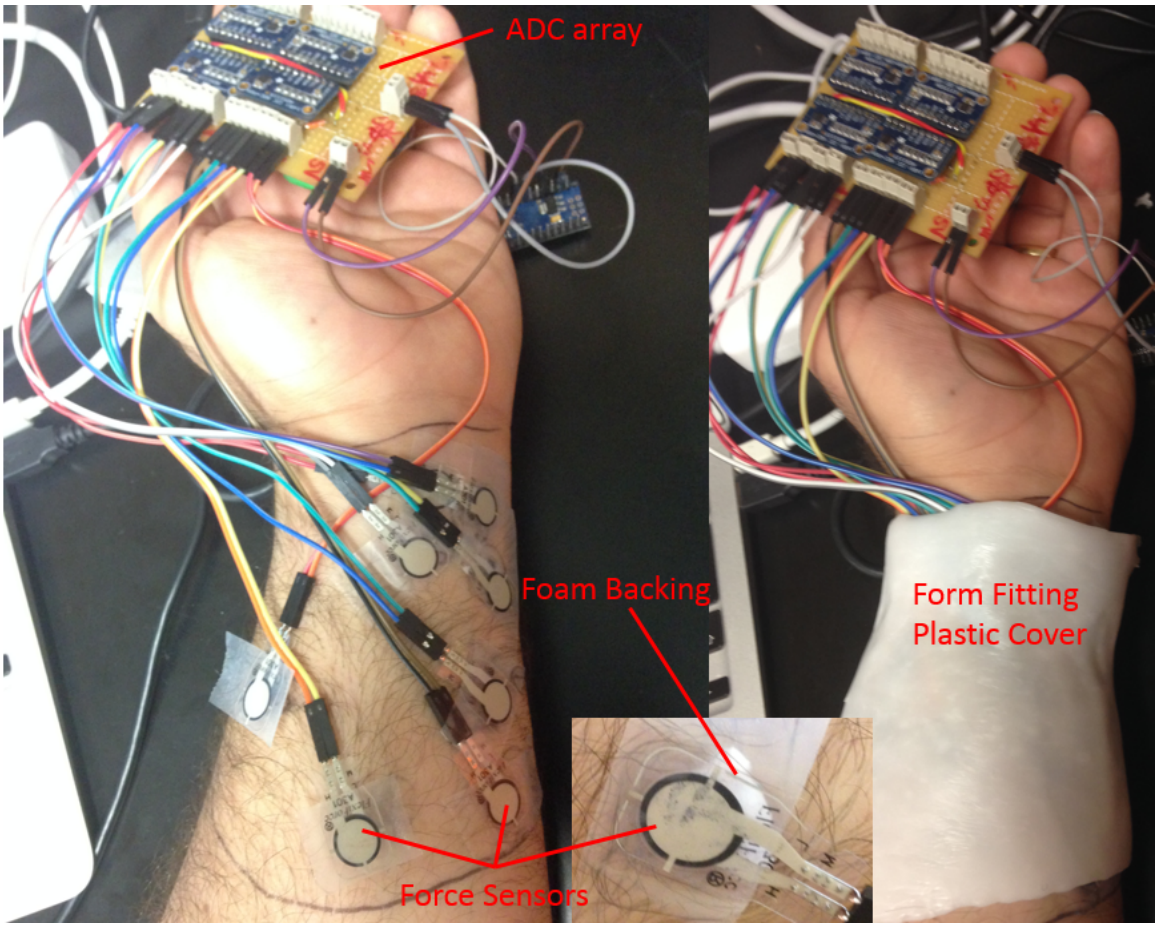

Figure 5.17 Force Myography Setup.

5.2.5 FMG Results

The data capture from 8 channels of FMG can be seen in Fig. 5.18. The channels represent the raw Analog to Digital Converter output, representing the pressure on the sensors. Dotted line represents the ground truth, which is the number of the

digit the subject was instructed to flex: 1 representing the thumb, 2 the index finger, etc. with 0 representing no digit flexion.

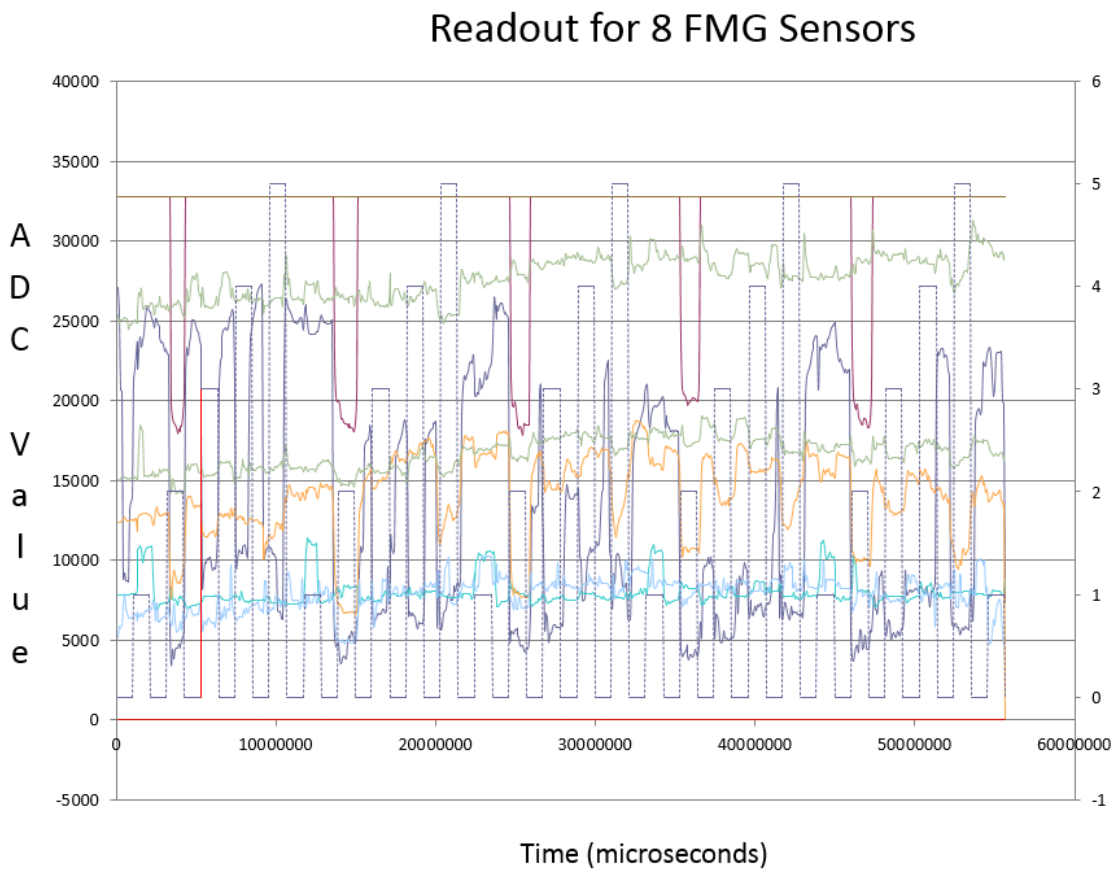

Figure 5.18 Eight channels of FMG.

It may be more instructive to examine one of the channels in which the FMG sensor placement was correct, as seen in Fig 5.19. It can be clearly seen from the figure that moving the index finger causes a decrease in the signal of the chosen force sensor.

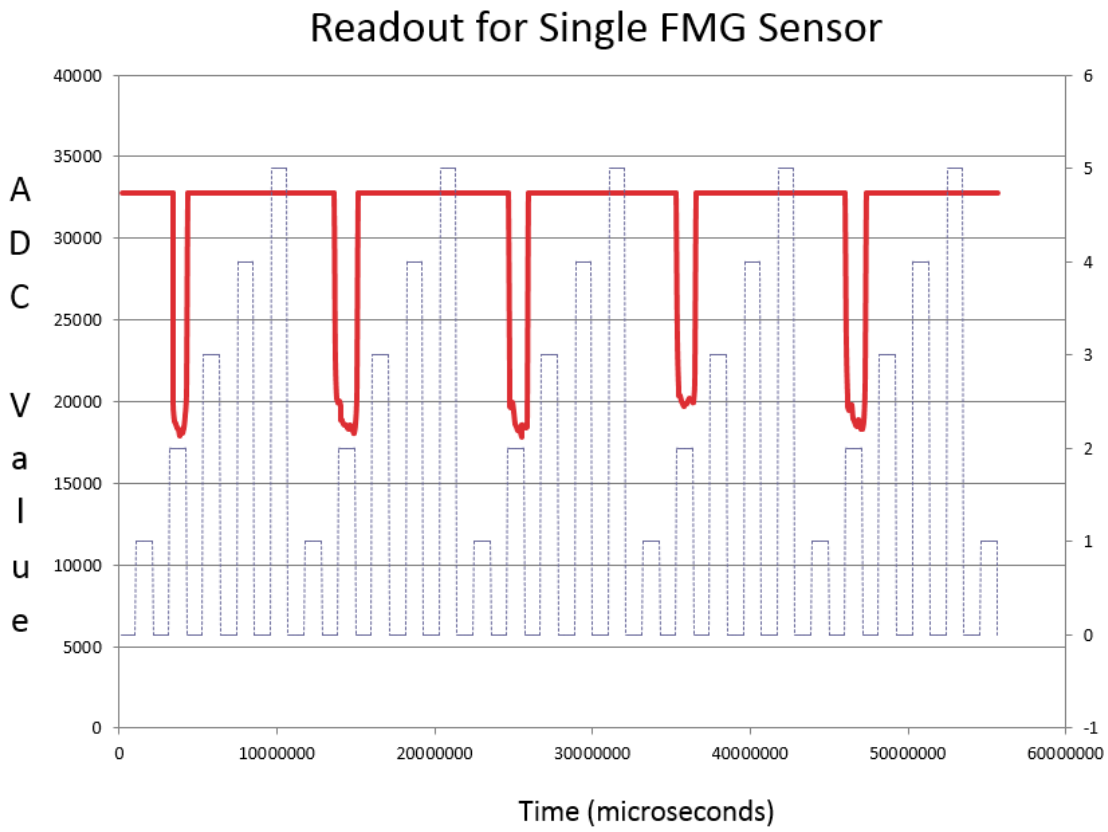

Figure 5.19 A single isolated channel of FMG.

5.3 Classification Results from FMG

5.3.1 FMG Data Import and Preprocessing

The subject was instructed follow an LED prompter to flex or relax each digit on the hand while both glove and EMG data was being acquired. The subject repeated the sequence of flashing LEDs (which were turned on for each digit at 3 second intervals) for at least 3 cycles. The experiment was repeated five times. Two datasets were picked for training and validation. The first dataset can be seen in Fig. 5.20.

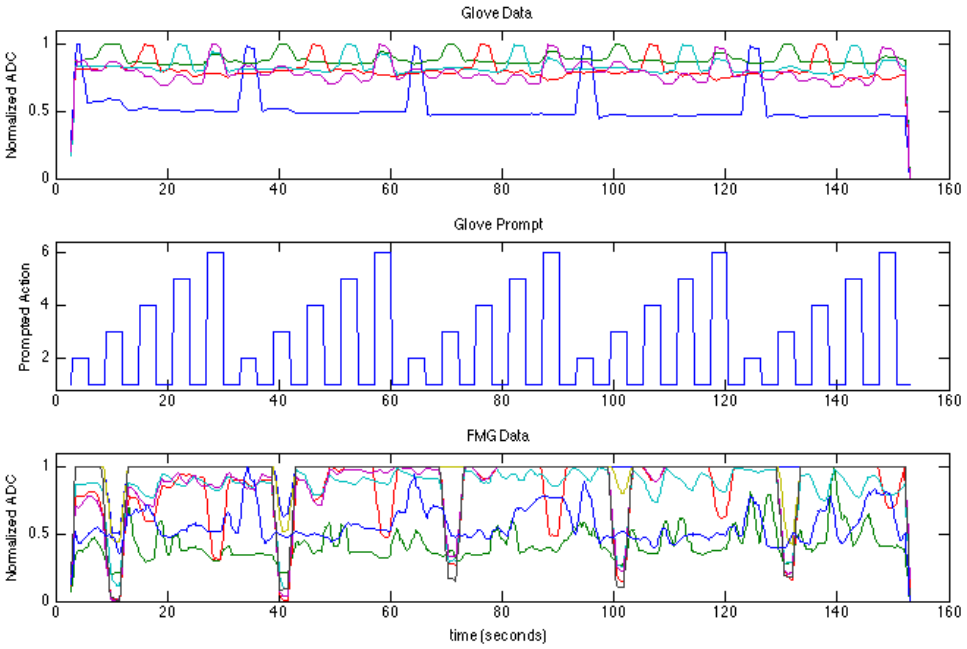

Figure 5.20 Imported Data from FMG and Glove Experiment.

After import, all data vectors (five channels of glove data, one channel of LED prompter data, and eight channels of force sensor data) were run through a moving average filter with a window size of 10.

At each step of the classification experiment, the vector representing the ground truth data and the vector obtained as user intent were compared using a root mean squared (RMS) calculation.

5.3.2 Obtaining User Intent from Glove Data

In this experiment, the data logged from the LED prompter system is assumed to be the ground truth. The first task involved training a neural network to classify

the glove data using the ground truth. The ground truth data are logged as numbers from 1 to 5, with the following meanings:

1. Perform no action
2. Flex thumb
3. Flex index finger
4. Flex middle finger
5. Flex ring finger
6. Flex ring finger

The neural network trained in this step was used in the subsequent steps to provide user intent data to the FMG classifier.

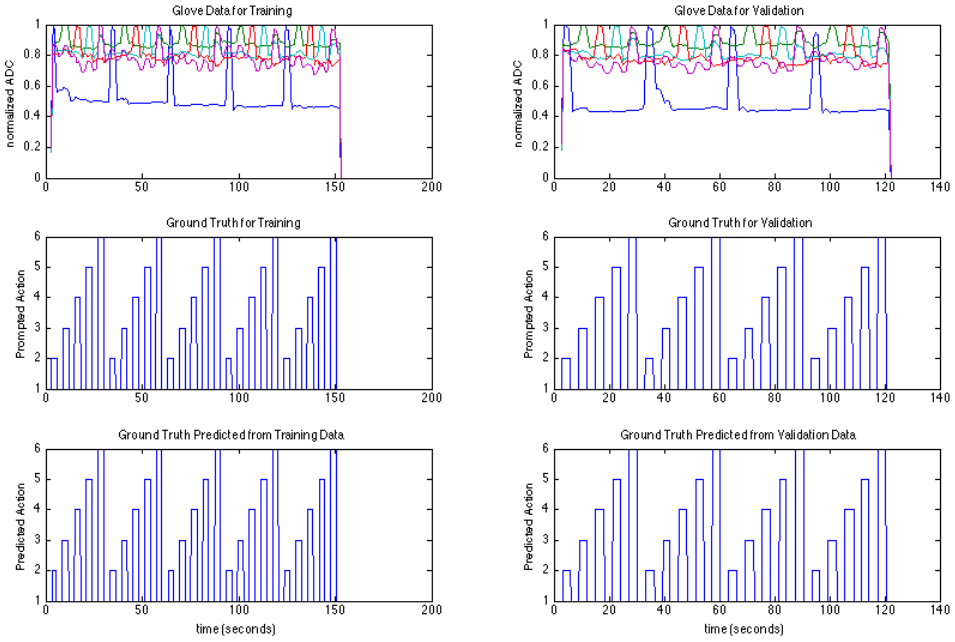

Figure 5.21 Classification of glove data by GPNN for creating ground truth for subsequent steps.

Fig. 5.21 shows the glove data used for training (upper left), as well as the glove data used for validation from a different run of the experiment (upper right). In the middle left is displayed the ground truth as logged by the LED prompter system. The glove data were classified using one of the built-in classification algorithms for MATLAB's Neural Network Toolbox (patternnet) with a hidden layer size of 30 and an output size of 6 (Fig. 5.22). In the bottom left is the output of the classifier validated with the glove data used for training. The figure in the right middle shows the ground truth logged by the LED prompter for the second run of the experiment. In the bottom right is the validation output which is the output of the network when classifying a different glove dataset (top right). The system was able to perform this classification with very few errors (RMS distance between the ground truth (middle left) and the output predicted from the training set (bottom left) was 23.82 and the RMS distance between the ground truth of the second dataset (middle right) and the output of the classifier (bottom right) was 25.22).

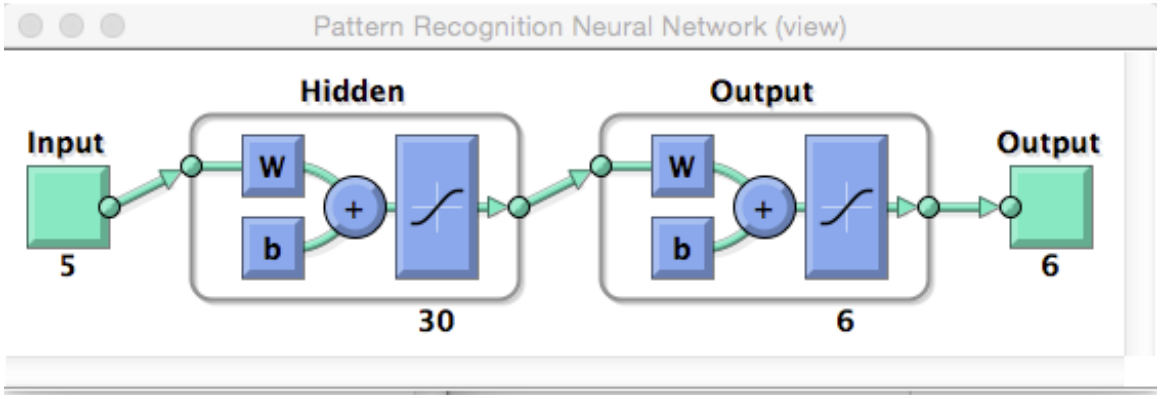

Figure 5.22 Visual representation of the neural network which uses glove data (GPNN) in order to create ground truth for subsequent steps of the classification experiment.

Fig. 5.23 displays preliminary data demonstrating the ability of the system to predict user intent from FMG data alone when trained on the ground truth from the LED prompter system.

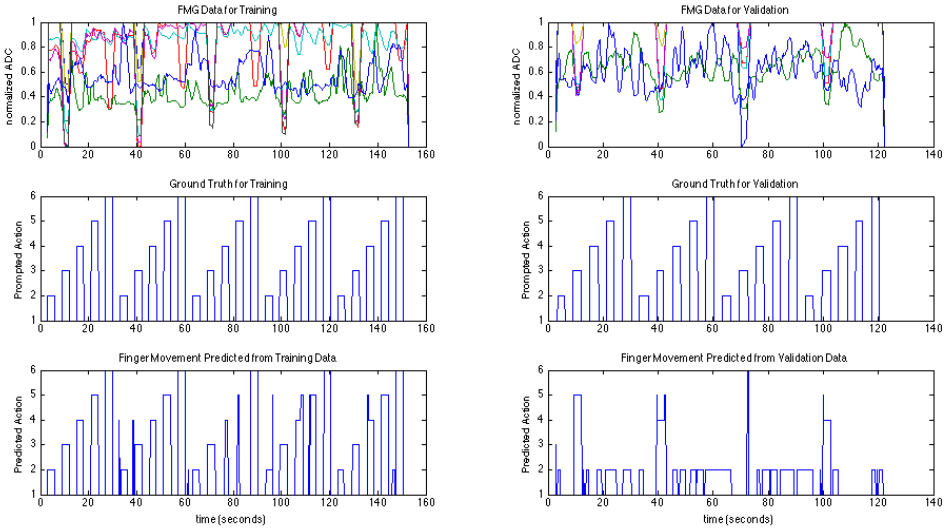

Figure 5.23 FMG based finger movement prediction by UIPNN (trained using ground truth).

On the top left of Fig. 5.23 is the FMG input dataset obtained from eight FlexiForce (TekScan Inc., USA) sensors. The vertical axis displays the output from the analog to digital converter (ADC) normalized to be between 0 and 1. The middle left displays the ground truth used to train the FMG Neural Network classifier. Bottom left displays the prediction of the neural network classifier trained on the ground truth when processing the training input. On the right top is the validation data from a different FMG experiment run on the same subject (with roughly the same sensor placement). The middle right displays the ground truth from the second experiment, and the bottom right displays the output of the neural network classifier

when run on the second dataset. When run on the new (validation) data, the system was able to predict some (but not all) of the finger movements. This can be improved with a longer training time and better sensor placement. The RMS error between the ground truth and the output of the network from the training set was 28.35 vs. the RMS error between the ground truth and the output of the network from the validation data which was 59.64.

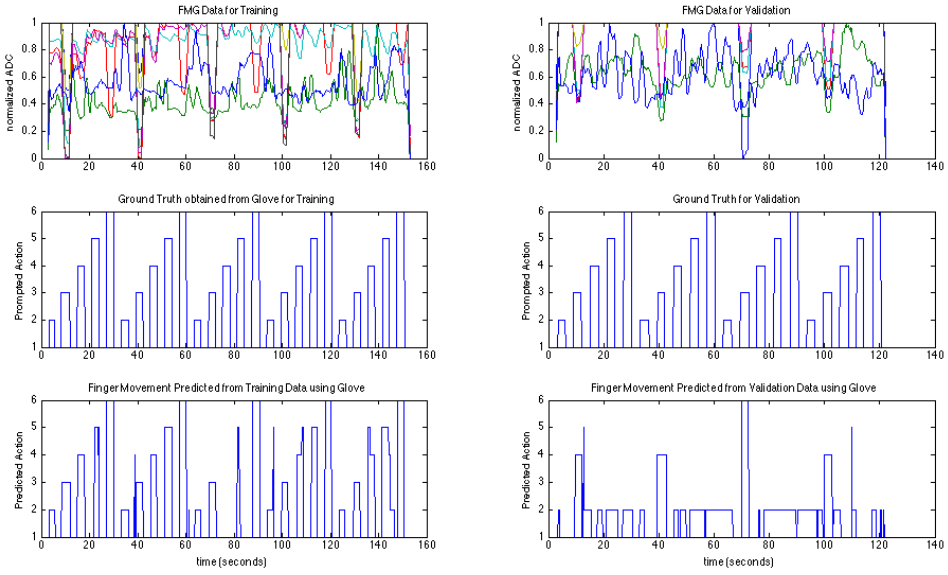

Figure 5.24 FMG based finger movement prediction by UIPNN (trained using glove data processed by GPNN).

Fig. 5.24 shows the end result of the experiment. It is very similar to Fig. 5.23 except that the data used as the training set (left middle and right middle) was obtained from classifying the glove input data by the network discussed in Fig. 5.21 (the GPNN). Although the prediction from the validation set was not highly accurate, its accuracy and shape mirrors that of the prediction of the neural network

trained on actual ground truth (Fig. 5.23). The RMS error between the ground truth and the Finger movement predicted from the training data (left bottom, Fig. 5.24) was 36.68, and the RMS error between the ground truth of the validation data and the predicted output was 58.83 (these values are very similar to the values obtained from the UIPNN trained on actual ground truth data, which were 28.35 and 59.64 respectively).

5.4 Discussion

The preliminary data presented demonstrate that input from a kinematic tracking glove can be turned into ground truth which can subsequently be used to train a neural network classifier on FMG data. The muscle data and glove tracking data in the experiments listed in this section were obtained from the same arm. For an amputee, the system would be modified slightly in order to obtain tracking data from the intact arm and muscle data from the residual limb. This has been demonstrated in the context of EMG previously by Sebelius et al. [98]. We have demonstrated that the concept can be extended to FMG data.

It can be seen from the data above that classification can be achieved by training a neural network classification system with bilateral movements performed by a healthy person or the amputee. While it may be possible to get FMG data to give classification information by itself, this modality is probably best used as a complement to EMG. FMG has the advantage of not being affected by skin conductivity or sweat. FMG also has the advantage of being able to detect movement due to tendons where the EMG signal may not be very strong. It is, however, highly susceptible to mechanical noise.

A robust, field deployable EMG classification and training system should include the following elements:

1. A dense array of EMG and FMG sensors.
2. A kinematic tracking system which is easy for the amputee to put on. This may be a lightweight glove, a computer vision based system if the training is to be done in a specified location, or an advanced version of the fingernail worn sensors discussed earlier with the ability to localize themselves with respect to each other.
3. A clear process for calibration. In the case of the system discussed, this would involve making the same gestures on the tracked intact hand and the residual amputated limb which is instrumented with EMG and FMG sensors.
4. A way to track the arm position. While not the focus of this chapter, it is important for a training algorithm to have awareness of which position the arm is in. This will allow the user to utilize the designed system in different arm positions and have better performance.
5. An embedded computer on which to run the calculations. One option may be to have a wi-fi or 4G data equipped device which can offload the calculation to a more powerful server. Neural network training much more CPU intensive than neural network execution.

5.5 Future Work

5.5.1 EMG/FMG based Classifier Training

Even though the training of an FMG classifier using bilateral mirrored movements for input has been demonstrated, work is under way to compare the perfor-

mance of a classifier based on FMG alone, EMG alone and a combination of FMG and EMG. Work is also under way to increase the number of FMG sensors by using an FMG sensor array, thus obviating the need for careful sensor placement. The current hypothesis is that the combination of FMG and EMG will yield better classification performance.

5.5.2 Kinematic Tracking based on Fingernail Device

The fingernail based gesture tracking modality discussed can be extended with some effort to be a kinematic tracker by embedding additional light sensors and accelerometers into the fingernail based sensors. This may necessitate communication between the fingernail sensors with a non-light-based modality (such as BlueTooth) in which relative intensities of light seen by each fingernail based device are compared to derive a kinematic model of the hand. This, in turn, would allow the fingernail based trackers to be used as a kinematic input device (similar to the previously discussed Mirroring Glove) without the user having to put on an additional device. Combining all the modalities discussed in this work should render the system more usable in its entirety to the end user.

CHAPTER 6

CONCLUSION

In Aim 1, an alternative method of prosthetic device control was presented which may enable patients who are currently unable to utilize robotic prosthetic devices to finally be able to use them, using intact hand based controls. Construction details of the Mirroring Glove, Gesture Glove, and the lightweight Fingernail Device were also presented in Aim 1.

In Aim 2, an implementation of using bilateral movements and hand-tracking to train an EMG classifier was presented and extended the concept to FMG and combined EMG/FMG data.

6.1 Discussion

For Aim 1, the preliminary results indicate that intact hand based control can be as fast as, and in some cases faster than, EMG based control. For Aim 2, the feasibility of using a portable gesture tracking system to train EMG and FMG classification systems has been demonstrated, suggesting that a hybrid system using both EMG and FMG data may be even more useful.

Each system presented in this work has been tested on its own in order to refine the system and demonstrate its standalone usefulness. It appears from the preliminary data that while device control based on mirroring movements on the prosthetic device is useful for certain tasks, the scheme of using discrete input gestures, as in the case

of the Gesture Glove and the Fingernail Device, is quicker to learn and less error prone. This limitation stems in part from the fact that the state of the art prosthetic device used in the experiments (Touch Bionics robo-limb) was never designed for teleoperation. The only commands supported by the device are "open" and "close" and each command takes around 1.4 seconds to complete. Furthermore, the device is an open-loop control system without the ability to encode finger flexion. The open loop nature of the device, along with the time lag between the issuance of a command via the Mirroring Glove and the completion of the grasp pattern seems to frustrate some users. This can be improved in the future by designing prosthetic devices which afford closed loop control.

6.2 Future Work

Feasibility of a classifier training system based on intact hand tracking for FMG has been demonstrated. Work remains on refining this system, including the fusion of EMG/FMG data and validation with amputees performing activities of daily living with the resulting system.

For the Fingernail Device, work remains on miniaturization and refinement, including the incorporation of wireless capability. An easy refinement to this system would be the addition of double-tap, triple-tap, and long-tap gestures, which would greatly increase the types of commands that can be mapped to the device.

Another exciting possibility for the Fingernail Device is the use of this device for 5 or 6 DOF kinematic hand tracking in addition to discrete gesture input. This may be possible with the incorporation of additional LED emitters and light sensors on each fingernail, in which the entirety of the light intensity dataset would be used for inferring the current hand pose. Such a device may potentially be useful for non-

prosthetic uses, including the control of teleoperated robots and the decoding of sign language.

The work presented in this thesis in Aim 1 (healthy limb based control) and Aim 2 (EMG/FMG classifier training based on kinematic tracking of the intact hand) is meant to be complementary and synergistic.

6.2.1 Envisioned System

The final system which realizes the work presented is a fully wearable Human Robot Interface for both commanding the prosthetic device and training the EMG/FMG classifier. In an ideal scenario, the user would use the fingernail mounted devices to switch EMG recognition modes, switch between EMG and intact hand based control, activate the training mode, train the system, and put the system back into normal use mode.

In order to realize its full potential, the envisioned system needs to be prototyped and refined in collaboration with amputees who are given an active role in the development process.

APPENDIX A
SOURCE CODE FOR EMBEDDED UNIX SERVER AND MIRRORING GLOVE SYSTEM

This appendix presents the source code for the Mirroring Glove System

A.1 Arduino Code

```
1  //Oguz Yetkin
2  //oyetkin@gmail.com
3  //Control of a TouchBionics robo-limb prosthetic device with a ...
       Multi FIngered Glove.
4  //Requires handcontrol2 running on the Linux computer driving the ...
       prosthetic device via a USB to CAN adapter (we used KVaser ...
       Leaf 2)
5
6  #include <Servo.h>
7
8  //calibration values
9  //0 for normal operation
10 //1 for raw values
11 //2 for processed values
12 //3 for processed value dump
13
14 const int calibration = 0;
15 boolean autocalibration = false;
16
17 boolean isPaused = false;
18 boolean thumbdebug = true;
19
20 const int PAUSE_LED = 10;
21 const int CALIB_LED = 11;
22 const int CALIB_LED_2 = 9;
23 const int OUTPUT_LED = 3;
24 const int THUMB_CLOSE_LED = 6;
25
26 void togglePause(){
27
28   if(isPaused == true){
29     isPaused = false;
30     digitalWrite(PAUSE_LED, LOW);
31     Serial.println("#UNPAUSED");
32   }else if(isPaused == false){
33     isPaused = true;
34     digitalWrite(PAUSE_LED,HIGH);
35     Serial.println("#PAUSED");
36   }
37 }
38
39
40 float mapfloat(long x, long in_min, long in_max, long out_min, ...
       long out_max)
41 {
```

```
42   return (float)(x - in_min) * (out_max - out_min) / ...
         (float)(in_max - in_min) + out_min;
43 }
44
45
46 const int CALIB_BUTTON_PIN = 7;
47 const int PAUSE_BUTTON_PIN = 2;
48
49
50 int nreads = 7;  //number of analogRead's per channel to average
51
52 //const int printdelay = 1500;
53 const int printdelay = 10;
54
55 //for compare()
56 //how much a value has to change before the change is taken seriously
57 float DELTA =4;
58
59
60 const int nPins = 5;
61
62 boolean doPrint = false;
63
64
65 Servo myservos[nPins];  // create servo object to control a servo
66                 // a maximum of eight servo objects can be created
67
68 const int analogInPins[] = {A0,A1,A2,A3,A4};
69 const int LEDPins[] = {11, 9, 6, 5, 3,10};
70
71
72 //7/1/2016 Oguz Glove v3
73
74 int sensorMaxs[] = {409,153,252,255,398};  //min finger flexion, ...
       determined experimentally
75 int sensorMins[] = {309,85,157,149,219};  //sensor values at max ...
       finger flexion, det
76
77 //determine whether to open or close the digit
78 int oldDigitValues[] = {0,0,0,0,0};
79 int newDigitValues[] = {0,0,0,0,0};
80
81 //OY 01/29/2015
82 int low_threshold[] = {40,20,40,40,40};
83 int high_threshold[] = {60,40,60,60,60};
84
85
86 //These are the decimal CANBus Mailbox IDs for the TouchBionics ...
       robo-limb prosthetic hand
87 //the handControl2 program uses the syntax:
```

```
88 // ./handcontrol2 DIGIT_ID PERCENT_CLOSED DIGIT_ID PERCENT_CLOSED ...
       ... for each digit
89 //262 thumb rotator (not used right now -- set to closed as of ...
       1/28/2015 for DMD paper manipulation task)
90
91 //257 thumb
92 //258 index
93 //259 mid
94 //260 ring
95 //261 little
96
97 //CO OY 1/28/2015 mapping thumb instead of thumb rotator to 0th ...
       encoder
98 //const int digitIds[] = {262,258,259,260,261};
99
100 const int digitIds[] = {257, 258, 259, 260, 261};
101
102 //uncomment two following lines to read actual values
103 //const int sensorMins[] = {0,0,0};  //min finger flexion, ...
       determined experimentally
104 //const int sensorMaxs[] = {1023,1023,1023};  //sensor values at ...
       max finger flexion, determined experimentally
105
106
107 const int servoPins[] = {9,10,11,6,5};
108 const int analogOutPin = 9; // Analog output pin that the LED is ...
       attached to
109
110 int sensorValues[] = {0,0,0,0,0};        // value read from the pot
111 int outputValues[] = {0,0,0,0,0};        // value output to the ...
       PWM (analog out)
112 int oldOutputValues[] = {0,0,0,0,0};
113
114 void setup() {
115   // initialize serial communications at 9600 bps:
116   Serial.begin(9600);
117   //Serial.begin(115200);
118
119   //commenting out servo code OY 3/29/2015
120   /*
121   for(int i=0;i< nPins ;i++){
122     myservos[i].attach(servoPins[i]);
123   }
124   */
125   pinMode(CALIB_BUTTON_PIN,INPUT);
126   pinMode(PAUSE_BUTTON_PIN,INPUT);
127
128
129   for(int i=0;i<nPins+1;i++){
130     pinMode(LEDPins[i], OUTPUT);
131     //analogWrite(LEDPins[i],0);
```

```
132     digitalWrite(LEDPins[i],LOW);
133     delay(30);
134   }
135   //analogReference(INTERNAL);
136 }
137
138 boolean compare(float val1, float val2, float Δ){
139   if(abs(val1-val2)>Δ){
140     return true;
141   }else{
142     return false;
143   }
144 }
145
146 void loop() {
147   // read the analog in value:
148   //Serial.println("");
149
150   //OY 8/27/2014
151
152   int calibpin = digitalRead(CALIB_BUTTON_PIN);  //calibpin is ...
        pulled up, so 0 means autocalibration should be true
153   int pausepin = digitalRead(PAUSE_BUTTON_PIN);
154   if(calibpin == 1){
155     autocalibration = true; // to be reset after calibration
156   }
157   if(pausepin == 1){
158     togglePause();
159   }
160
161   if(autocalibration){
162     Serial.println("reading maxs");
163     digitalWrite(CALIB_LED, HIGH);
164     for(int i=0;i<nPins;i++){
165       //take an average -- perhaps this should be a moving average
166       sensorValues[i] = 0;
167       for(int j=0;j<nreads*4;j++){
168         sensorValues[i] += analogRead(analogInPins[i]);
169         sensorMins[i] = -1;
170         delay(10);
171       }
172       sensorValues[i] /= nreads*4;
173     }
174
175     for(int i=0;i<nPins;i++){
176       sensorMaxs[i] = sensorValues[i];
177       Serial.print(sensorMaxs[i]);
178       Serial.print(",");
179     }
180     digitalWrite(CALIB_LED, LOW);
181     delay(30);
```

```
182     digitalWrite(CALIB_LED_2, HIGH);
183     Serial.println("get ready to do sensor mins in 2 sec");
184     delay(2000);
185     Serial.println("flex each digit to its max");
186
187     for(int i=0;i<nPins;i++){
188       //take an average -- perhaps this should be a moving average
189       sensorValues[i] = 0;
190       for(int j=0;j<nreads*4;j++){
191         sensorValues[i] += analogRead(analogInPins[i]);
192         delay(10);
193       }
194       sensorValues[i] /= nreads*4;
195     }
196
197    for(int i=0;i<nPins;i++){
198       sensorMins[i] = sensorValues[i];
199       Serial.print(sensorMins[i]);
200       Serial.print(",");
201     }
202     Serial.println("autocalibration done");
203     digitalWrite(CALIB_LED_2, LOW);
204     autocalibration = false;
205   }//end autocalibration
206
207
208   for(int i=0;i<nPins;i++){
209       //take an average -- perhaps this should be a moving average
210       sensorValues[i] = 0;
211       for(int j=0;j<nreads;j++){
212         sensorValues[i] += analogRead(analogInPins[i]);
213         delay(10);
214       }
215       sensorValues[i] /= nreads;
216
217     if(0 == calibration){
218       outputValues[i] = ...
              constrain(mapfloat(sensorValues[i],sensorMins[i],sensorMaxs[i],0,100),0,100
219
220       if(compare(oldOutputValues[i],outputValues[i],DELTA)){
221         if((outputValues[i] < low_threshold[i])){
222           //newDigitValues[i] = 0;
223           newDigitValues[i] = 100;
224         }
225         if(outputValues[i] > high_threshold[i]){
226           //newDigitValues[i] = 100;
227           newDigitValues[i] = 0;
228         }
229       }
230       oldOutputValues[i] = outputValues[i];
231       //output
```

```
232
233
234       if(oldDigitValues[i] != newDigitValues[i]){
235         doPrint = true;
236         oldDigitValues[i] = newDigitValues[i];
237       }
238     }// end output block
239     if(1 == calibration){
240           //calib
241       outputValues[i] = sensorValues[i];
242       Serial.print(" outputValues[");
243       Serial.print(i);
244       Serial.print("]");
245       Serial.print("=");
246       Serial.print(outputValues[i]);
247       if(i == nPins - 1){
248         int calibpin = digitalRead(CALIB_BUTTON_PIN);
249         Serial.print("c: ");
250         Serial.print(calibpin);
251         Serial.println();
252         Serial.print("pin13: ");
253         Serial.println(digitalRead(13));
254         Serial.print(" pin8: ");
255         Serial.println(digitalRead(8));
256         Serial.print(" pin12: ");
257         Serial.println(digitalRead(12));
258         Serial.print(" pin2: ");
259         Serial.println(digitalRead(2));
260         Serial.print(" pin7: ");
261         Serial.println(digitalRead(7));
262       }
263     }
264     if(2 == calibration){
265           //calib
266       outputValues[i] = ...
              constrain(mapfloat(sensorValues[i],sensorMins[i],sensorMaxs[i],0,100),0,100
267       Serial.print(" outputValues[");
268       Serial.print(i);
269       Serial.print("]");
270       Serial.print("=");
271       Serial.print(outputValues[i]);
272     }
273
274
275     if(3 == calibration){
276       outputValues[i] = ...
              constrain(mapfloat(sensorValues[i],sensorMins[i],sensorMaxs[i],0,100),0,100
277
278       //regular print of 5 values
279
280       Serial.print(100 - outputValues[i]);
```

```
281         if(i != nPins - 1){ //don't print the last comma
282           Serial.print(",");
283         }
284 
285       }
286 
287 
288      myservos[i].write(outputValues[i]);
289 
290     }//end for
291 
292 
293 
294 
295     //print block for handcontrol2
296     if(doPrint && ! isPaused){
297       digitalWrite(OUTPUT_LED, HIGH);
298       Serial.print("./handcontrol2 ");
299       for(int i=0;i<nPins;i++){
300         Serial.print(digitIds[i]);
301          Serial.print(" ");
302          Serial.print(newDigitValues[i]);
303          Serial.print(" ");
304 
305          if(thumbdebug){
306            if(i == 0 && newDigitValues[i] == 100){
307              digitalWrite(THUMB_CLOSE_LED,HIGH);
308              delay(200);
309              digitalWrite(THUMB_CLOSE_LED,LOW);
310            }
311          }
312       }
313       //CO OY 1/28/2015 for now, manually put in thumb closure
314       //Serial.print(" 257 0 ");
315       //OY 1/28/2015 setting thumb rotator to closed and mapping ...
              thumb encoder to thumb closure instead
316       Serial.print(" 262 100 ");
317       doPrint = false;
318       Serial.println();
319       delay(printdelay);
320       digitalWrite(OUTPUT_LED, LOW);
321     }
322 
323 
324 
325     if(calibration > 0){
326      Serial.println();
327    }
328    delay(2);
329 
330  }
```

A.2 Source Code for CANBus driver Utility handcontrol2

This code uses the Linux Drivers for KVaser Leaf USB to CAN adapter device.

```
1  /*
2  **                 Copyright 2012 by Kvaser AB, Mlndal, Sweden
3  **                         http://www.kvaser.com
4  **
5  ** This software is dual licensed under the following two licenses:
6  ** BSD-new and GPLv2. You may use either one. See the included
7  ** COPYING file for details.
8  **
9  ** License: BSD-new
10 ** ...
   ==========================================================================
11 ** Redistribution and use in source and binary forms, with or without
12 ** modification, are permitted provided that the following ...
   conditions are met:
13 **     * Redistributions of source code must retain the above ...
   copyright
14 **       notice, this list of conditions and the following ...
   disclaimer.
15 **     * Redistributions in binary form must reproduce the above ...
   copyright
16 **       notice, this list of conditions and the following ...
   disclaimer in the
17 **       documentation and/or other materials provided with the ...
   distribution.
18 **     * Neither the name of the <organization> nor the
19 **       names of its contributors may be used to endorse or ...
   promote products
20 **       derived from this software without specific prior ...
   written permission.
21 **
22 ** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND ...
   CONTRIBUTORS "AS IS" AND
23 ** ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED ...
   TO, THE IMPLIED
24 ** WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR ...
   PURPOSE ARE
25 ** DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY
26 ** DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR ...
   CONSEQUENTIAL DAMAGES
27 ** (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE ...
   GOODS OR SERVICES;
28 ** LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) ...
   HOWEVER CAUSED AND
29 ** ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT ...
   LIABILITY, OR TORT
30 ** (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF ...
   THE USE OF THIS
```

```
31 ** SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 **
33 **
34 ** License: GPLv2
35 ** ...
     ============================================================================
36 ** This program is free software; you can redistribute it and/or
37 ** modify it under the terms of the GNU General Public License
38 ** as published by the Free Software Foundation; either version 2
39 ** of the License, or (at your option) any later version.
40 **
41 ** This program is distributed in the hope that it will be useful,
42 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
43 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
44 ** GNU General Public License for more details.
45 **
46 ** You should have received a copy of the GNU General Public License
47 ** along with this program; if not, write to the Free Software
48 ** Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  ...
     02110-1301, USA.
49 **
50 ** ...
     ----------------------------------------------------------------------------
51 **/
52
53 /*
54  * Kvaser Linux Canlib
55  * Send a CAN message
56  */
57
58 #include <canlib.h>
59 #include <stdio.h>
60 #include <signal.h>
61 #include <errno.h>
62 #include <unistd.h>
63
64 unsigned char msg[4] = {0};
65 int digits[6] = {262,258,259,260,261,257};
66
67 void check (char* id, canStatus stat)
68 {
69   char buf[50];
70
71   buf[0] = '\0';
72   canGetErrorText(stat, buf, sizeof(buf));
73   if (stat != canOK) {
74     printf("%s: failed, stat=%d (%s)\n", id, (int)stat, buf);
75   } else {
76     //printf("%s: OK\n", id);
77   }
78 }
```

```
79  /* OY 8/28/2014 adding stopAll to be issued before anything else */
80
81  void stopAll(canHandle h){
82      int i = 0;
83      msg[0] = 0;
84      msg[2] = 1; /* msg[2] and msg[3] = {1,41} specify max PWM for ...
            RoboLimb */
85      msg[3]  = 41;
86      for(i=0;i<6;i++){
87          msg[1] = 0;
88          check("canWrite", canWrite(h, digits[i], msg, 4, 0));
89      }
90  }
91
92  /*
93   * Send messages until ctrl-c is pressed
94   */
95
96  int main (int argc, char *argv[])
97  {
98    canHandle h;
99    int channel;
100   int bitrate = BAUD_1M;
101
102   errno = 0;
103   if (argc != 13 || (channel = 0, errno) != 0) {
104     printf("usage %s address byte\n", argv[0]);
105     exit(1);
106   } else {
107     printf("Sending a message on channel %d\n", channel);
108   }
109
110
111   /* Allow signals to interrupt syscalls(e.g in canReadBlock) */
112   siginterrupt(SIGINT, 1);
113
114   /* Open channel, set parameters and go on bus */
115
116   //h = canOpenChannel(channel, canOPEN_EXCLUSIVE | ...
          canOPEN_REQUIRE_EXTENDED);
117   h = canOpenChannel(channel, canOPEN_EXCLUSIVE);
118   if (h < 0) {
119     printf("canOpenChannel %d failed\n", channel);
120     return -1;
121   }
122
123   canBusOff(h);
124   check("canSetBusParams", canSetBusParams(h, bitrate, 4, 3, 1, ...
          1, 0));
125   // Work-around for Leaf bug
```

```
126   check("canSetBusOutputControl", canSetBusOutputControl(h, ...
          canDRIVER_NORMAL));
127   check("canBusOn", canBusOn(h));
128   //Sumit Modified to run Robolimb
129   //08/27/2014
130   msg[0] = 0;
131   msg[2] = 1;
132   msg[3] = 41;
133
134   /* OY 8/28/2014 stop all digits first */
135   stopAll(h);
136
137   if (atoi(argv[2]) == 0)
138     msg[1] = 2;
139   if (atoi(argv[2]) == 100)
140     msg[1] = 1;
141
142   check("canWrite", canWrite(h, atoi(argv[1]), msg, 4, 0));
143
144   if (atoi(argv[4]) == 0)
145     msg[1] = 2;
146   if (atoi(argv[4]) == 100)
147     msg[1] = 1;
148
149   check("canWrite", canWrite(h, atoi(argv[3]), msg, 4, 0));
150
151   if (atoi(argv[6]) == 0)
152     msg[1] = 2;
153   if (atoi(argv[6]) == 100)
154     msg[1] = 1;
155
156   check("canWrite", canWrite(h, atoi(argv[5]), msg, 4, 0));
157
158   if (atoi(argv[8]) == 0)
159     msg[1] = 2;
160   if (atoi(argv[8]) == 100)
161     msg[1] = 1;
162
163   check("canWrite", canWrite(h, atoi(argv[7]), msg, 4, 0));
164
165   if (atoi(argv[10]) == 0)
166     msg[1] = 2;
167   if (atoi(argv[10]) == 100)
168     msg[1] = 1;
169
170   check("canWrite", canWrite(h, atoi(argv[9]), msg, 4, 0));
171
172   if (atoi(argv[12]) == 0)
173     msg[1] = 2;
174   if (atoi(argv[12]) == 100)
175     msg[1] = 1;
```

```
176
177     check("canWrite", canWrite(h, atoi(argv[11]), msg, 4, 0));
178
179
180     check("canWriteSync", canWriteSync(h, 1000));
181
182
183     //check("canBusOff", canBusOff(h));
184     check("canClose", canClose(h));
185
186     return 0;
187 }
```

APPENDIX B
SOURCE CODE FOR GESTURE GLOVE SYSTEM

This appendix presents the source code for the Gesture Glove System. The UNIX and CANBus side of the code are the same for all three systems.

B.1 Arduino Code

The code listed below uses the Arduino Statistics library written by Scott Daniels https://github.com/provideyourown/statistics

```
/*

GloveControlConductievPinch.ino
Oguz Yetkin 2/17/2015 oyetkin@gmail.com
Code for SPIE STA 2015 conference in Baltimore, reads conductive ...
    thread gesture device
via analog inputs, decides on a gesture

Based on AnalogReadSerial example, which is in the public domain.

 */


#include <Servo.h>

//calibration values
//0 for normal operation
//1 for raw values
//2 for processed values
//3 pinch glove mode -- direct thresholding

const int pinch_glove_threshold = 15;

const int calibration = 1;
//const int printdelay = 1500;
const int printdelay = 150;

//for compare()
//how much a value has to change before the change is taken seriously
float DELTA =8;
int nreads = 8;

float analogNread(int AI, int n){

  float val = 0;
  for(int i=0;i<nreads;i++){
    val += analogRead(AI);
    delay(20);
  }
  val/=nreads;
  return val;
```

```
41 }
42
43 const int nPins = 5;
44
45 boolean doPrint = false;
46
47
48 Servo myservos[nPins];  // create servo object to control a servo
49                         // a maximum of eight servo objects can ...
                               be created
50
51
52 // These constants won't change.  They're used to give names
53 // to the pins used:
54 const int analogInPins[] = {A0,A1,A2,A3,A5};  // Analog input pin ...
      that the potentiometer is attached to
55
56 const int sensorMaxs[] = {20,20,20, 20 ,20};  //min finger ...
      flexion, determined experimentally
57 const int sensorMins[] = {0,0,0,0,0};  //sensor values at max ...
      finger flexion, determined experimentally
58
59
60 //determine whether to open or close the digit
61 int oldDigitValues[] = {0,0,0,0,0};
62 int newDigitValues[] = {0,0,0,0,0};
63 //const int low_threshold = 40;
64 //const int high_threshold = 70;
65
66 int low_threshold[] = {40,40,40,40,40};
67 int high_threshold[] = {70,70,70,70,70};
68
69 //const int digitIds[] = {101,102,103};
70 //OY Decimal values for direct serial control
71 //const int digitIds[] = {257,258,259,};
72
73 //257 thumb
74 const int digitIds[] = {262,258,259,260,261};
75
76 const int servoPins[] = {9,10,11,6,5};
77
78
79 const int analogOutPin = 9; // Analog output pin that the LED is ...
      attached to
80
81 int sensorValues[] = {0,0,0,0,0};        // value read from the pot
82 int outputValues[] = {0,0,0,0,0};        // value output to the ...
      PWM (analog out)
83 int oldOutputValues[] = {0,0,0,0,0};
84
85 void setup() {
```

```
86    // initialize serial communications at 9600 bps:
87    Serial.begin(9600);
88    for(int i=0;i< nPins ;i++){
89      myservos[i].attach(servoPins[i]);
90    }
91  }
92
93  boolean compare(float val1, float val2, float Δ){
94    if(abs(val1-val2)>Δ){
95      return true;
96    }else{
97      return false;
98    }
99  }
100
101 void loop() {
102
103   for(int i=0;i<nPins;i++){
104     sensorValues[i] = analogNread(analogInPins[i],nreads);
105   //thumb
106
107     if(0 == calibration){
108       outputValues[i] = ...
              constrain(map(sensorValues[i],sensorMins[i],sensorMaxs[i],0,100),0,100);
109       if(compare(oldOutputValues[i],outputValues[i],DELTA)){
110         if((outputValues[i] < low_threshold[i])){
111           //newDigitValues[i] = 0;
112           newDigitValues[i] = 100;
113         }
114         if(outputValues[i] > high_threshold[i]){
115           //newDigitValues[i] = 100;
116           newDigitValues[i] = 0;
117         }
118       }
119       oldOutputValues[i] = outputValues[i];
120       //output
121
122
123       if(oldDigitValues[i] != newDigitValues[i]){
124       //if(compare(oldDigitValues[i],newDigitValues[i],25)){
125         doPrint = true;
126
127         oldDigitValues[i] = newDigitValues[i];
128       }
129
130     }// end output block
131     if(1 == calibration){
132           //calib
133       outputValues[i] = sensorValues[i];
134       Serial.print(" outputValues[");
135       Serial.print(i);
```

```
136         Serial.print("]");
137         Serial.print("=");
138         Serial.print(outputValues[i]);
139       }
140       if(2 == calibration){
141             //calib
142         outputValues[i] = ...
              constrain(map(sensorValues[i],sensorMins[i],sensorMaxs[i],0,100),0,100);
143         Serial.print(" outputValues[");
144         Serial.print(i);
145         Serial.print("]");
146         Serial.print("=");
147         Serial.print(outputValues[i]);
148       }
149
150
151        if(3 == calibration){
152         //outputValues[i] = ...
              constrain(map(sensorValues[i],sensorMins[i],sensorMaxs[i],0,100),0,100);
153         outputValues[i] = sensorValues[i];  //use raw sensor values
154         //LEFTOFF need oldOutputValues OY 8/27/2014
155         if(compare(oldOutputValues[i],outputValues[i],DELTA)){
156           if((outputValues[i] < pinch_glove_threshold)){
157             //newDigitValues[i] = 0;
158             newDigitValues[i] = 100;
159           }
160           if(outputValues[i] ≥ pinch_glove_threshold){
161             //newDigitValues[i] = 100;
162             newDigitValues[i] = 0;
163           }
164         }
165         oldOutputValues[i] = outputValues[i];
166         //output
167
168
169         if(oldDigitValues[i] != newDigitValues[i]){
170         //if(compare(oldDigitValues[i],newDigitValues[i],25)){
171           doPrint = true;
172           oldDigitValues[i] = newDigitValues[i];
173         }
174
175       }// end output block pinch glove
176       myservos[i].write(outputValues[i]);
177
178     }//end for
179
180
181     //print block for handcontrol2
182     if(doPrint){
183       Serial.print("./handcontrol2 ");
184       for(int i=0;i<nPins;i++){
```

```
185         Serial.print(digitIds[i]);
186          Serial.print(" ");
187          Serial.print(newDigitValues[i]);
188          Serial.print(" ");
189       }
190       //for now, manually put in thumb closure
191       Serial.print(" 257 0 ");
192       doPrint = false;
193       Serial.println();
194       delay(printdelay);
195     }
196 
197     if(calibration > 0){
198      Serial.println();
199    }
200    delay(2);
201 
202 }
```

APPENDIX C
SOURCE CODE FOR FINGERNAIL SENSOR SYSTEM

In this appendix, we present the source code for the Fingernail Mounted System. The UNIX and CANBus side of the code are the same for all three systems.

C.1 Multifrequency Multiple Flasher for Testing

```
1  #include <Adafruit_ADS1015.h>
2
3  #include <Wire.h>
4
5  Adafruit_ADS1115 ads;  /* Use this for the 16-bit version */
6  const int ledPin =  12;      // the number of the LED pin
7
8  // Variables will change :
9  int ledState = LOW;             // ledState used to set the LED
10 int test;
11
12 // Generally, you should use "unsigned long" for variables that ...
       hold time
13 // The value will quickly become too large for an int to store
14 unsigned long previousMillis = 0;        // will store last time ...
       LED was updated
15 int pulse_duration=100;
16 // constants won't change
17 const long interval = 1000+pulse_duration;// interval at which to ...
       blink (milliseconds)
18
19 void setup() {
20   // set the digital pin as output:
21   pinMode(ledPin, OUTPUT);
22   Serial.begin(115200);
23   // ads.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit ...
         = 3mV      0.1875mV (default)
24   // ads.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit ...
         = 2mV      0.125mV
25    //ads.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit ...
          = 1mV      0.0625mV
26   // ads.setGain(GAIN_FOUR);       // 4x gain   +/- 1.024V  1 bit ...
         = 0.5mV    0.03125mV
27   //ads.setGain(GAIN_EIGHT);      // 8x gain   +/- 0.512V  1 bit ...
         = 0.25mV   0.015625mV
28    ads.setGain(GAIN_SIXTEEN);    // 16x gain  +/- 0.256V  1 bit = ...
          0.125mV  0.0078125mV
29  ads.begin();
30 }
31
32 void loop() {
33 int16_t adc0;
34 adc0 = ads.readADC_SingleEnded(0);
```

```
35   unsigned long currentMillis = millis();
36
37   if (currentMillis - previousMillis ≥ interval) {
38     // save the last time you blinked the LED
39     previousMillis = currentMillis;}
40
41     // if the LED is off turn it on and vice-versa:
42     if((currentMillis-previousMillis≥0)&&(currentMillis-previousMillis≤pulse_duration
43     {ledState = HIGH;
44       test=1;
45     }
46     else ...
          if((currentMillis-previousMillis>pulse_duration)&&(currentMillis-previousMill
47     {ledState = LOW;
48       test=0;
49     }
50     digitalWrite(ledPin, ledState);
51
52
53     // set the LED with the ledState of the variable:
54
55
56   //int sensorValue = analogRead(A3);
57   // print out the value you read:
58   //Serial.print(sensorValue);
59   //Serial.print(",");
60   //Serial.println(test);
61    Serial.print(micros());
62    Serial.print(",");
63    Serial.print(adc0);
64    Serial.print(",");
65    Serial.println(test);
66    //Serial.println(currentMillis-previousMillis);
67
68 }
```

C.2 MATLAB Code Extract Response Delay

```
1  function [ response_delays ] = extract_response_delay( t, prompt, ...
       resp )
2  %extract_response_delay returns vector of response delays
3  %   Given a time vector, a vector prompt representing prompts , and
4  %   a vector resp representing responses, returns time delays
5  %Oguz Yetkin 3/17/2016 for SPIE Paper fingernail sensor testing
6
7      search_window = 1000000; %1 second
8      response_delays=[];
9      prompt_times = [];
```

```
    response_times = [];
    prompt_found = 0;
    for i=2:1:length(t)

        if (prompt(i) > 0) && (prompt(i-1) == 0)
            prompt_found = 1;
            prompt_times = [prompt_times t(i)];
        end
        if prompt_found && (prompt(i-1) >0) && (prompt(i) == 0)
            prompt_found = 0;
            prompt_time = t(i);
        end
        if prompt_found
            if resp(i) > 0
                if not (prompt(i) == resp(i))
                    disp('ERROR: incorrect response encountered')
                else
                    resp_delay = t(i) - prompt_times(end);
                    response_delays = [response_delays resp_delay];
                    response_times = [response_times t(i)];
                    prompt_found = 0;
                end
            end
        end
    end
        %debug, show result visually

        fig=figure;
        hax=axes;

        plot(t,prompt,'red'); ylim([0 2.4]);xlabel 'time (s)'
        hold on
        plot(t,resp,'blue');


        x=t;
        hold on
        %plot(x,sin(x))
        for i=1:length(prompt_times)
            SP=prompt_times(i); %your point goes here
            line([SP SP],get(hax,'YLim'),'Color',[1 0 0])
        end
        for i=1:length(response_times)
            SP=response_times(i); %your point goes here
            line([SP SP],get(hax,'YLim'),'Color',[0 0 1])
        end


end
```

C.3 FFT MATLAB Code

C.3.1 myfft.m

```
function [ frequencies, amplitudes ] = myfft( ...
    signal,Fs,window_begin,window_end )
%myfft Custom FFT Function by Oguz Yetkin
%   This function takes a signal, sampling rate, and as well as ...
    window_begin and window_end which
%   allow the user to exclude some frequencies from the output


T = 1/Fs;                       % Sample time
L = length(signal);                     % Length of signal
t = (0:L-1)*T;                % Time vector
t= t(1:length(t));
x = signal;

y=x;

NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(y(window_begin:window_end),NFFT)/L;

%Create the properly sized vectors for the frequencies and ...
    amplitudes Oguz OY 2/19/2016
frequencies = Fs/2*linspace(0,1,NFFT/2+1);
amplitudes = 2*abs(Y(1:NFFT/2+1)); %Single-Sided Amplitude ...
    Spectrum of

end
```

C.3.2 FFT_Based_Fingernail_Detector.m

```
clear all;
close all;

%determined by the dataset
%for skipbegin and skipend
lowest_freq_of_interest = 15;
highest_freq_of_interest = 75;

%numdata=csvread('fingernail_31_31_short.csv');

%this works 3/18/2016
%numdata=csvread('fingernail_20_90.csv');

%works but with aliasing
numdata=csvread('fingernail_41_75_twofinger_prompt.csv');
```

```
16
17 %numdata=csvread('fingernail_20_90_twofinger.csv');
18
19 t=numdata(:,1);%load column 1 to matlab
20 amp=numdata(:,2);%load column 2 to matlab
21 ground_truth=numdata(:,3);
22 %Fs=785;
23 Fs=800;
24 %Fs=1800;
25 L = length(amp);
26 P=amp;
27 %finish code import
28
29
30 resultvector = [];
31 resultvector_amp = [];
32 subplot(5,1,1);
33 plot(t,P);
34 title 'original time series'
35 subplot(5,1,2);
36 [freq,ampl] = myfft(amp,Fs,1,length(amp));
37
38
39 lowest_freq_of_interest_idx = ...
      floor((lowest_freq_of_interest/(0.5*Fs))*length(freq));
40 highest_freq_of_interest_idx = ...
      floor((highest_freq_of_interest/(0.5*Fs))*length(freq));
41
42
43 winwidth = 200;
44 winstep=20;
45
46 %these values work for 13 Hz 2/26/2016
47 %?begin = 100;
48 %skipend = length(ampl)-3000;
49
50
51 %values to encompass [13,80] Hz, determined
52 %experimentally from plot of (freq,ampl) and the index of freq
53 %this displays the correct ranges (13 and 53 Hz) but is not clear
54 %skipbegin = 170;
55 %skipend = 5000;
56
57 skipbegin = lowest_freq_of_interest_idx;
58 skipend = highest_freq_of_interest_idx;
59
60 %threshold determined from teh maximum and minimum amplitudes in ...
      FFT'd window
61 filtered_signal_threshold = 0.065;
62
63 plot(freq, ampl);
```

```
64 title 'fft, full time series'
65 for i=1:winstep:length(P)-winwidth
66     winbegin = i;
67     winend = i+winwidth;
68     [freq, ampl] = myfft(P, Fs,winbegin,winend);
69     %find the index of the maximum amplitude, then find the
70     %maximum frequency which corresponds to it;
71
72     %OY 02/24/2016 the ampl vector returned seems to have a spurious
73     %spike in index 1, also at the end which was causing the ...
          entire function
74     %to not work.
75     %Introduced skipbegin and skipend OY 02/26/2016
76     maximum_frequency = freq(find (ampl == ...
          max(ampl(skipbegin:skipend))));
77     maximum_amplitude = ampl(find (ampl == ...
          max(ampl(skipbegin:skipend))));
78     %determining filtered_signal_threshold during the loop did ...
          not work,
79     %this needs to be determined once the intensity of the signal of
80     %interest (after FFT) is known when touching and not touching
81     %
82     %filtered_signal_threshold = 0.5*(maximum_amplitude - ...
          ampl(find (ampl == max(ampl(skipbegin:skipend)))));
83     %disp('filtered_signal_threshold: ');
84     %disp(filtered_signal_threshold);
85     if maximum_amplitude > filtered_signal_threshold;
86         resultvector = [resultvector maximum_frequency];
87     else
88         resultvector = [resultvector 0];
89     end
90     resultvector_amp = [resultvector_amp maximum_amplitude];
91     %plot(freq, ampl)
92     %title('Single-Sided Amplitude Spectrum of y(t)')
93     %xlabel('Frequency (Hz)')
94     %ylabel('|Y(f)|')
95     %pause(0.1);
96 end
97 subplot(5,1,3);
98 plot(resultvector);
99 title('maximum freq vs samples');
100 subplot(5,1,4);
101 plot(ground_truth,'red');
102 title('ground truth')
103 subplot(5,1,5);
104 plot(resultvector_amp);
105 title('amplitudes at maximum freq')
106 b=ones(30644,1);
107
108 figure(2)
109 subplot(3,1,1)
```

```
yy = smooth(amp)
plot(yy)
subplot(3,1,2)
yy2 = smooth(ampl)
plot(yy2)
subplot(3,1,3)
yy3 = smooth(resultvector)
plot(yy3)

%figure for publication


figure;
t_sec = t/1000000;
subplot(3,1,1);
plot(t_sec,P);
set(gca,'FontSize',10)
xlabel 'time (s)'
set(gca,'FontSize',12)
ylabel 'amplitude'
set(gca,'FontSize',14)
title 'original time series'
subplot(3,1,2);
t_resultvector = ...
    linspace(min(t_sec),max(t_sec),length(resultvector));
plot(t_resultvector,resultvector);
set(gca,'FontSize',10)
xlabel 'time (s)'
set(gca,'FontSize',12)
ylabel 'max freq'
set(gca,'FontSize',14)
title('maximum freq vs samples');
subplot(3,1,3)
%set(gca,'FontSize',18)
plot(t_sec,ground_truth,'red');
set(gca,'FontSize',10)
xlabel 'time (s)'
set(gca,'FontSize',12)
ylabel 'max freq'
set(gca,'FontSize',14)
title('ground truth')
```

APPENDIX D
SOURCE CODE EMG AND FMG CLASSIFICATION

In this appendix, we present the source code for the EMG and FMG classification systems created.

D.1 Arduino Code to Acquire LED Prompter, Glove, and FMG Data

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1115 ads0(0x48); // ADR-> GND
Adafruit_ADS1115 ads1(0x49); // ADR -> VDD
Adafruit_ADS1115 ads2(0x4A); // ADR -> SDA
Adafruit_ADS1115 ads3(0x4B); // ADR -> SCL
//Adafruit_ADS1015 ads;     /* Use thi for the 12-bit version */

//Oguz Yetkin OY 7/2/2016.
int master = 1;
int valueread = 0;
int waittime = 2;  //time to pause between reads in ms, for sync
int triggervalue = 0;
int oldtriggervalue = 0;

int suppress_extra_info = 0;

long curr_micros = 0;
long old_micros = 0;


int flexNow = 0;  //0 relax, 1 idx 2 mid 3 ring 4 pinkie 5 thumb
int flexNowIndex = 0;
const int nFlex = 10;
int flexSequence[nFlex] = {0, 1, 0, 2, 0, 3, 0, 4, 0, 5};
int flexLEDs[6] = {0, 13, 12, 11, 10, 9}; // idx, mid, ring, ...
    pinkie, thumb
//OY 2/16/2016
int sensorValues[6] = {0, 0, 0, 0, 0, 0};
int pulse_pin = 5;
//long interval = 3000000;
long interval = 3000000;



void setup(void)
{
  flexNowIndex = 0;
  pinMode(13, OUTPUT);  //flash 13 while reading value for debug
  pinMode(12, OUTPUT);  //flash 13 while reading value for debug
  pinMode(11, OUTPUT);  //flash 13 while reading value for debug
  pinMode(10, OUTPUT);  //flash 13 while reading value for debug
```

```
43   pinMode(9, OUTPUT);  //flash 13 while reading value for debug
44
45   digitalWrite(13, LOW);
46   digitalWrite(12, LOW);
47   digitalWrite(11, LOW);
48   digitalWrite(10, LOW);
49   digitalWrite(9, LOW);
50
51   pinMode(7, OUTPUT);
52   pinMode(8, INPUT_PULLUP);
53   if (0 == master) {
54     digitalWrite(13, LOW);
55   }
56   digitalWrite(7, HIGH);
57
58   Serial.begin(115200);
59   //Serial.println("Hello!");
60
61   //Serial.println("Getting single-ended readings from AIN0..3");
62   //Serial.println("ADC Range: +/- 6.144V (1 bit = 3mV/ADS1015, ...
         0.1875mV/ADS1115)");
63
64   // The ADC input range (or gain) can be changed via the following
65   // functions, but be careful never to exceed VDD +0.3V max, or to
66   // exceed the upper and lower limits if you adjust the input range!
67   // Setting these values incorrectly may destroy your ADC!
68   // ...
                                                                      ...
         ADS1015  ADS1115
69   // ...
                                                                      ...
         -------  -------
70   // ads0.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 ...
         bit = 3mV      0.1875mV (default)
71   // ads0.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 ...
         bit = 2mV      0.125mV
72
73   /*
74     ads0.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit ...
           = 1mV      0.0625mV
75     ads1.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit ...
           = 1mV      0.0625mV
76     ads2.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit ...
           = 1mV      0.0625mV
77     ads3.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit ...
           = 1mV      0.0625mV
78   */
79   ads0.setGain(GAIN_FOUR);        // 16x gain  +/- 0.256V  1 bit ...
       = 0.125mV  0.0078125mV
80   ads1.setGain(GAIN_FOUR);        // 16x gain  +/- 0.256V  1 bit ...
       = 0.125mV  0.0078125mV
```

```
81    //GAIN_ONE for glove
82    ads2.setGain(GAIN_ONE);         // 16x gain  +/- 0.256V  1 bit = ...
          0.125mV  0.0078125mV
83    ads3.setGain(GAIN_ONE);         // 16x gain  +/- 0.256V  1 bit = ...
          0.125mV  0.0078125mV
84
85    // ads0.setGain(GAIN_FOUR);        // 4x gain   +/- 1.024V  1 ...
          bit = 0.5mV    0.03125mV
86    // ads0.setGain(GAIN_EIGHT);       // 8x gain   +/- 0.512V  1 ...
          bit = 0.25mV   0.015625mV
87    // ads0.setGain(GAIN_SIXTEEN);     // 16x gain  +/- 0.256V  1 ...
          bit = 0.125mV  0.0078125mV
88
89    ads0.begin();
90  }
91
92  void read_and_print_values() {
93    int16_t adc[16];
94    // Read the ADC and put it to a variable name
95    // clean up for efficiency
96    byte sensor = 0;
97    //for(byte loop = 0; loop < 4; loop++){
98    for (byte x = 0; x < 4; x++) {
99      adc[sensor] = ads0.readADC_SingleEnded(x);
100     sensor++;
101   }
102   for (byte x = 0; x < 4; x++) {
103     adc[sensor] = ads1.readADC_SingleEnded(x);
104     sensor++;
105   }
106   for (byte x = 0; x < 4; x++) {
107     adc[sensor] = ads2.readADC_SingleEnded(x);
108     sensor++;
109   }
110   for (byte x = 0; x < 4; x++) {
111     adc[sensor] = ads3.readADC_SingleEnded(x);
112     sensor++;
113   }
114   if (0 == suppress_extra_info) {
115     Serial.print(curr_micros);
116     Serial.print(",");
117     Serial.print(flexNow);
118     Serial.print(",");
119   }
120   for (byte x = 0; x < 16; x++) {
121     Serial.print(adc[x]);
122     Serial.print(",");
123   }
124   Serial.println(" ");
125 }
126
```

```
127
128 void loop(void)
129 {
130   curr_micros = micros();
131   long time_elapsed = curr_micros - old_micros;
132   if (time_elapsed > interval) {
133     old_micros = curr_micros;
134     flexNowIndex++;
135     flexNowIndex %= nFlex; // go to the next state
136     flexNow = flexSequence[flexNowIndex];
137
138     //OY 11/17/2015 hard-coding values for now
139     if (flexNow == 0) {
140       digitalWrite(13, LOW);
141       digitalWrite(12, LOW);
142       digitalWrite(11, LOW);
143       digitalWrite(10, LOW);
144       digitalWrite(9, LOW);
145       digitalWrite(pulse_pin, HIGH);
146       //analogWrite(pulse_pin, 0);
147     } else {
148       digitalWrite(13, LOW);
149       digitalWrite(12, LOW);
150       digitalWrite(11, LOW);
151       digitalWrite(10, LOW);
152       digitalWrite(9, LOW);
153       digitalWrite(flexLEDs[flexNow], HIGH);
154       digitalWrite(pulse_pin, LOW);
155       //analogWrite(pulse_pin, 128);
156     }
157   }//end prompting code
158       if (1 == master) {
159       digitalWrite(7, LOW); //low on receiving Arduino will ...
              trigger read
160       //digitalWrite(13, HIGH); //debug
161       delay(waittime);
162       read_and_print_values();
163       valueread = 1;
164       digitalWrite(7, HIGH);  //high suppresses output on second ...
              arduino
165     } else {
166       oldtriggervalue = triggervalue;
167       triggervalue = !digitalRead(8);
168       //Serial.println(triggervalue);
169       //trigger read only on rising edge
170       if (0 == valueread && 1 == triggervalue && 0 == ...
              oldtriggervalue) {
171         //if(0 == valueread && 1 == triggervalue);
172         digitalWrite(13, HIGH); //debug
173         delay(waittime);
174         read_and_print_values();
```

```
175           valueread = 1;
176           digitalWrite(13, LOW);
177         }
178         if ((1 == oldtriggervalue) && (0 == triggervalue)) {
179           //reset valueread, prime for next read
180           valueread = 0;
181         }
182       }//end else
183   }
```

D.2 MATLAB Code for Classification and Training

D.2.1 run_FMG.m

This is the main code for FMG classification and training discussed in Chapter 5. It depends on the following codes, also listed in this appendix: normalize.m, importPromptGLoveFMG.m, plotPromptGLoveFMG.m.

```
1  %run_FMG.m
2  % Oguz Yetkin 7/4/2016
3  %This script written for the experiments on 7/4/2016 but can be ...
       used for
4  %similar data
5  %
6  %import, plot, and train on FMG experiment
7  % %Plots produced: 1) Ground truth1, Ground truth predicted from ...
       Glove,
8  % Ground truth 2, Ground truth predicted from another run of the ...
       glove
9  %                   3) Finger movements predicted from FMG using ...
       Ground Truth
10 %                      as training data
11 %                   4) Finger movements predicted from FMG using ...
       Glove as
12 %                      Training Data
13 %
14 %Experiment setup: LED prompter, Glove, and FMG data acquired
15 %
16 %column 1: Time in ms
17 %Column 2: LED prompter data.  0 = no action, 1 index finger .. 5 ...
       thumb
18 %(will be translated to 1 = no action, 2 = index inger .. 6 = ...
       thumb in
19 %order to be more MATLAB friendly for the ind2vec function
20 %Column 3-10: FMG data from FlexiForce pressure sensors
21 %Column 11-15: Glove data (high means open, low means flexed, ...
       max=2^15
22
23 filename1='data2A.csv'; %filename1 for training
```

```
filename2='data2B.csv'; %filename2 for verification


%filename1='data2.csv'; %filename1 for training
%filename2='data3.csv'; %filename2 for verification


%ground_truth1 is ground truth data from prompter
%ngData1 is normalized glove data
%neData1 is FMG (Force Myography) data from pressure sensors on ...
   the arm
[gTime1, ground_truth1, ngData1, neData1] = ...
   importPromptGloveFMG(filename1);
[gTime2, ground_truth2, ngData2, neData2] = ...
   importPromptGloveFMG(filename2);
figure;
plotPromptGloveFMG(gTime1, ground_truth1, ngData1, neData1);

%turn ground truth data into a training set matrix in which every row
%represents a class.  Class 1 = no action, Class 2 = thumb ...
   flexion .. Class 6 =
%little finger flexion
training_set1 = full(ind2vec(ground_truth1'));
gloveNet1 = patternnet(30);
gloveNet1.trainParam.showWindow = false;
gloveNet1.trainParam.showCommandLine = true;
gloveNet1.trainParam.min_grad = 1e-9; %default 1e-6
gloveNet1.trainParam.max_fail = 20; %default 5

gloveNet1 = train(gloveNet1, ngData1', training_set1);
%run the neural network on the original dataset
ground_truth_from_glove1 = vec2ind(gloveNet1(ngData1'));
%run the neural network on a different dataset (cross-validation)
ground_truth_from_glove2 = vec2ind(gloveNet1(ngData2'));
figure;
subplot(3,2,1);
plot(gTime1, ngData1);
ylabel 'normalized ADC'
title 'Glove Data for Training'
subplot(3,2,3);
plot(gTime1, ground_truth1);
title 'Ground Truth for Training'
ylabel 'Prompted Action'
subplot(3,2,5);
plot(gTime1, ground_truth_from_glove1);
title 'Ground Truth Predicted from Training Data'
ylabel 'Predicted Action'
xlabel 'time (seconds)'
%right side of graph
subplot(3,2,2);
plot(gTime2, ngData2);
```

```
ylabel 'normalized ADC'
title 'Glove Data for Validation'
subplot(3,2,4);
plot(gTime2, ground_truth2);
title 'Ground Truth for Validation'
ylabel 'Prompted Action'
subplot(3,2,6);
plot(gTime2, ground_truth_from_glove2);
title 'Ground Truth Predicted from Validation Data'
ylabel 'Predicted Action'
xlabel 'time (seconds)'

disp('RMS error between ground truth and predicted ground truth ...
    from training set for Glove');
disp(sqrt(sum((ground_truth1-ground_truth_from_glove1').^2)));
disp('RMS error between ground truth and predicted ground truth ...
    from validation set for Glove');
disp(sqrt(sum((ground_truth2-ground_truth_from_glove2').^2)));

%Now predict finger movements from FMG data using prompter as ...
    ground truth

%turn ground truth data into a training set matrix in which every row
%represents a class.  Class 1 = no action, Class 2 = thumb ...
    flexion .. Class 6 =
%little finger flexion
training_set1 = full(ind2vec(ground_truth1'));
FMGNet1_prompter = patternnet(30);
FMGNet1_prompter.trainParam.showWindow = false;
FMGNet1_prompter.trainParam.showCommandLine = true;
FMGNet1_prompter.trainParam.epochs = 1000; %default 1000
FMGNet1_prompter.trainParam.max_fail = 100; %default 5

FMGNet1_prompter = train(FMGNet1_prompter, neData1', training_set1);
%run the neural network on the original dataset
result_from_FMG1 = vec2ind(FMGNet1_prompter(neData1'));
%run the neural network on a different dataset (cross-validation)
result_from_FMG2 = vec2ind(FMGNet1_prompter(neData2'));

figure;
subplot(3,2,1);
plot(gTime1, neData1);
ylabel 'normalized ADC'
title 'FMG Data for Training'
subplot(3,2,3);
plot(gTime1, ground_truth1);
title 'Ground Truth for Training'
ylabel 'Prompted Action'
subplot(3,2,5);
plot(gTime1, result_from_FMG1);
title 'Finger Movement Predicted from Training Data'
```

```
ylabel 'Predicted Action'
xlabel 'time (seconds)'
%right side of graph
subplot(3,2,2);
plot(gTime2, neData2);
ylabel 'normalized ADC'
title 'FMG Data for Validation'
subplot(3,2,4);
plot(gTime2, ground_truth_from_glove2);
title 'Ground Truth for Validation'
ylabel 'Prompted Action'
subplot(3,2,6);
plot(gTime2, result_from_FMG2);
title 'Finger Movement Predicted from Validation Data'
ylabel 'Predicted Action'
xlabel 'time (seconds)'

disp('RMS error between ground truth and predicted Finger ...
    Movement from training set');
disp(sqrt(sum((ground_truth1-result_from_FMG1').^2)));
disp('RMS error between ground truth and predicted Finger ...
    Movement from validation set');
disp(sqrt(sum((ground_truth2-result_from_FMG2').^2)));

% GLOVE AS GROUND TRUTH

%Now predict finger movements from FMG data using glove as ground ...
    truth

%turn ground truth data into a training set matrix in which every row
%represents a class.  Class 1 = no action, Class 2 = thumb ...
    flexion .. Class 6 =
%little finger flexion
%training_set1 = full(ind2vec(ground_truth1'));
FMGNet1_glove = patternnet(30);
FMGNet1_glove.trainParam.showWindow = false;
FMGNet1_glove.trainParam.showCommandLine = true;
FMGNet1_glove.trainParam.epochs = 1000; %default 1000
FMGNet1_glove.trainParam.max_fail = 100; %default 5

FMGNet1_glove = train(FMGNet1_glove, neData1', ...
    ind2vec(ground_truth_from_glove1));
%run the neural network on the original dataset
result_from_FMG1_glove = vec2ind(FMGNet1_glove(neData1'));
%run the neural network on a different dataset (cross-validation)
result_from_FMG2_glove = vec2ind(FMGNet1_glove(neData2'));

figure;
subplot(3,2,1);
plot(gTime1, neData1);
ylabel 'normalized ADC'
```

```
164 title 'FMG Data for Training'
165 subplot(3,2,3);
166 plot(gTime1, ground_truth_from_glove1);
167 title 'Ground Truth obtained from Glove for Training'
168 ylabel 'Prompted Action'
169 subplot(3,2,5);
170 plot(gTime1, result_from_FMG1_glove);
171 title 'Finger Movement Predicted from Training Data using Glove'
172 ylabel 'Predicted Action'
173 xlabel 'time (seconds)'
174 %right side of graph
175 subplot(3,2,2);
176 plot(gTime2, neData2);
177 ylabel 'normalized ADC'
178 title 'FMG Data for Validation'
179 subplot(3,2,4);
180 plot(gTime2, ground_truth_from_glove2);
181 title 'Ground Truth for Validation'
182 ylabel 'Prompted Action'
183 subplot(3,2,6);
184 plot(gTime2, result_from_FMG2_glove);
185 title 'Finger Movement Predicted from Validation Data using Glove'
186 ylabel 'Predicted Action'
187 xlabel 'time (seconds)'
188
189 disp('RMS error between ground truth and predicted Finger ...
       Movement from training set');
190 disp(sqrt(sum((ground_truth1-result_from_FMG1_glove').^2)));
191 disp('RMS error between ground truth and predicted Finger ...
       Movement from validation set');
192 disp(sqrt(sum((ground_truth2-result_from_FMG2_glove').^2)));
193
194
195 disp('************** FMG Glove Experiment Summary **************');
196 disp('filename1 (training):');
197 disp(filename1);
198 disp('filename2 (validation):');
199 disp(filename2);
200
201 disp('RMS error between ground truth and predicted ground truth ...
       from training set for Glove');
202 disp(sqrt(sum((ground_truth1-ground_truth_from_glove1').^2)));
203 disp('RMS error between ground truth and predicted ground truth ...
       from validation set for Glove');
204 disp(sqrt(sum((ground_truth2-ground_truth_from_glove2').^2)));
205
206
207
208
209 disp('RMS error between ground truth and predicted Finger ...
       Movement from training set');
```

```
disp(sqrt(sum((ground_truth1-result_from_FMG1').^2)));
disp('RMS error between ground truth and predicted Finger ...
    Movement from validation set');
disp(sqrt(sum((ground_truth2-result_from_FMG2').^2)));


disp('RMS error between ground truth and predicted Finger ...
    Movement from training set');
disp(sqrt(sum((ground_truth1-result_from_FMG1_glove').^2)));
disp('RMS error between ground truth and predicted Finger ...
    Movement from validation set');
disp(sqrt(sum((ground_truth2-result_from_FMG2_glove').^2)));
```

D.2.2 normalize.m

This is a utility function.

```
function [ nvector ] = normalize( vector )
%Oguz Yetkin OY 7/4/2016 normalize each column to be a maximum of 1

    nvector = vector - repmat(min(vector), [length(vector) 1]);
    nvector = nvector ./ repmat(max(nvector), [length(nvector) 1]);


end
```

D.2.3 importPromptGloveFMG.m

This code imports FMG, glove, and LED prompter data.

```
function [gTime, gPrompt,ngData, neData] = ...
    importPromptGloveFMG(filename)
%importPromptGloveFMG
%Oguz Yetkin 7/4/2016
%
%input: filename
%outputs:   gTime = time in seconds
%           gPrompt: Ground Truth from LED prompter in MATLAB ...
    friendly format
%                   For use in classifier training
%                       %Class 1= no action
                        %Class 2 = thumb
                        %Class 3 = index
                        %Class 4 = middle
                        %Class 5 = ring
                        %Class 6 = little
%           ngData: normalized glove Data
```

```
%           neData: normalized EMG/FMG data


skipbegin = 20;  %empirically determined values to skip
skipend = 0;
conv_window = 10;

dataStruct=importdata(filename);
%data=dataStruct.data;
data=dataStruct;
%headers=dataStruct.colheaders;
gTime = data(skipbegin:(length(data)-skipend),1);
gTime = gTime/1000000.0; %turn time into seconds
eTime = gTime; %they are the same as on 7/3/2016 since Glove and
                    %FMG data were acquired from the same device
gPrompt = data(skipbegin:(length(data)-skipend),2);
gPrompt = gPrompt + 1; %MATLAB does not like 0 values.  gPrompt ...
    represents
                        %prompts given to the user.
                        %Class 1= no action
                        %Class 2 = thumb
                        %Class 3 = index
                        %Class 4 = middle
                        %Class 5 = ring
                        %Class 6 = little

gData = data(skipbegin:end,(11:15));
%complement the gData with 32768 to make sure unflexed means low, ...
    flexed
%means high
gData = 32768 - gData;
eData = data(skipbegin:end,(3:10)); %FMG data

%filter the data
s = size(gData);
ncols= s(2);
for i=1:ncols
    gData(:,i) = conv(gData(:,i), ones(1,conv_window),'same');
end
s = size(eData);
ncols= s(2);
for i=1:ncols
    eData(:,i) = conv(eData(:,i), ones(1,conv_window),'same');
end


gData = gData(1:(length(gData)-skipend),:);
eData = eData(1:(length(eData)-skipend),:);

ngData = normalize(gData);
neData =normalize(eData)
```

```
% subplot(3,1,1);
% plot(gTime,ngData);
% ylim([0 1.1])
% title('Glove Data')
% ylabel('Normalized ADC');
% subplot(3,1,2);
%
% plot(gTime,gPrompt);
% title('Glove Prompt')
% ylabel 'Prompted Action';
% ylim([0.8 6.4])
% subplot(3,1,3);
% plot(gTime,neData);
% ylim([0 1.1])
% ylabel('Normalized ADC');
% xlabel 'time (seconds)'
% title('FMG Data');
```

D.2.4 plotPromptGloveFMG.m

This code was used to generate the FMG plots in Chapter 5.

```
function  plotPromptGloveFMG(gTime, gPrompt, ngData, neData)
%plotPromptGloveFMG
%Oguz Yetkin 7/4/2016
%
%input: filename
%outputs:   gTime = time in seconds
%           gPrompt: Ground Truth from LED prompter in MATLAB ...
    friendly format
%                   For use in classifier training
%                       %Class 1= no action
                        %Class 2 = thumb
                        %Class 3 = index
                        %Class 4 = middle
                        %Class 5 = ring
                        %Class 6 = little
%           ngData: normalized glove Data
%           neData: normalized EMG/FMG data


subplot(3,1,1);
plot(gTime,ngData);
ylim([0 1.1])
title('Glove Data')
ylabel('Normalized ADC');
subplot(3,1,2);

```

```
27 plot(gTime,gPrompt);
28 title('Glove Prompt')
29 ylabel 'Prompted Action';
30 ylim([0.8 6.4])
31 subplot(3,1,3);
32 plot(gTime,neData);
33 ylim([0 1.1])
34 ylabel('Normalized ADC');
35 xlabel 'time (seconds)'
36 title('FMG Data');
```

REFERENCES

[1] IHSN, "Statistics on hand and arm loss." [Online]. Available: http://www.ishn.com/articles/97844-statistics-on-hand-and-arm-loss

[2] C. Toledo, L. Leija, R. Munoz, A. Vera, and A. Ramirez, "Upper limb prostheses for amputations above elbow: A review," *2009 Pan American Health Care Exchanges - PAHCE 2009*, no. April, pp. 104–108, 2009.

[3] G. W. V. Vidal, M. L. Rynes, Z. Kelliher, and S. J. Goodwin, "Review of Brain-Machine Interfaces Used in Neural Prosthetics with New Perspective on Somatosensory Feedback through Method of Signal Breakdown," *Scientifica*, vol. 2016, 2016.

[4] T. Yanagisawa, M. Hirata, Y. Saitoh, H. Kishima, K. Matsushita, T. Goto, R. Fukuma, H. Yokoi, Y. Kamitani, and T. Yoshimine, "Electrocorticographic control of a prosthetic arm in paralyzed patients," *Annals of neurology*, vol. 71, no. 3, pp. 353–361, 2012.

[5] T. Yanagisawa, M. Hirata, Y. Saitoh, T. Goto, H. Kishima, R. Fukuma, H. Yokoi, Y. Kamitani, and T. Yoshimine, "Real-time control of a prosthetic hand using human electrocorticography signals: technical note," *Journal of neurosurgery*, vol. 114, no. 6, pp. 1715–1722, 2011.

[6] J. Kubanek, K. J. Miller, J. G. Ojemann, J. R. Wolpaw, and G. Schalk, "Decoding flexion of individual fingers using electrocorticographic signals in humans," *Journal of neural engineering*, vol. 6, no. 6, p. 66001, 2009.

[7] X. Navarro, T. B. Krueger, N. Lago, S. Micera, T. Stieglitz, and P. Dario, "A critical review of interfaces with the peripheral nervous system for the control of

neuroprostheses and hybrid bionic systems," *Journal of the Peripheral Nervous System*, vol. 10, no. 3, pp. 229–258, 2005.

[8] TouchBionics, "Touch Bionics Mobile Apps." [Online]. Available: http://www.touchbionics.com/products/i-limb-mobile-apps

[9] D. Sueaseenak, T. Chanwimalueang, M. Sangworasil, and C. Pintavirooj, "An investigation of robustness in independent component analysis EMG," in *6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology 2009*, no. i, 2009, pp. 1102–1105. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5137237

[10] A. Alkan and M. Günay, "Identification of EMG signals using discriminant analysis and SVM classifier," *Expert Systems with Applications*, vol. 39, no. 1, pp. 44–47, 2012.

[11] P. K. Artemiadis and K. J. Kyriakopoulos, "An EMG-based robot control scheme robust to time-varying EMG signal features," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 3, pp. 582–588, 2010.

[12] J. A. Birdwell, L. J. Hargrove, T. A. Kuiken, and Others, "Extrinsic finger and thumb muscles command a virtual hand to allow individual finger and grasp control," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 1, pp. 218–226, 2015.

[13] D. R. Merrill, J. Lockhart, P. R. Troyk, R. F. Weir, and D. L. Hankin, "Development of an implantable myoelectric sensor for advanced prosthesis control," *Artificial organs*, vol. 35, no. 3, pp. 249–252, 2011.

[14] M. M. Lowery, R. F. H. Weir, and T. A. Kuiken, "Simulation of Intramuscular EMG Signal Detection using Implantable MyoElectric Sensors," in *Conference*

Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005. IEEE, 2005, pp. 360–363.

[15] R. F. Weir, P. R. Troyk, G. A. DeMichele, D. A. Kerns, J. F. Schorsch, and H. Maas, "Implantable myoelectric sensors (IMESs) for intramuscular electromyogram recording," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 159–171, 2009.

[16] R. F. Weir, P. R. Troyk, G. DeMichele, and T. Kuiken, "Implantable myoelectric sensors (IMES) for upper-extremity prosthesis control-preliminary work," in *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, vol. 2. IEEE, 2003, pp. 1562–1565.

[17] B. Schneiderman and C. Plaisant, "Designing the user interface," 1998.

[18] W. Joy and M. Horton, *An introduction to display editing with vi.* University of California Department of Electrical Engineering, 1980.

[19] V. Young, "Strategic UX: The Art of Reducing Friction." [Online]. Available: http://www.dtelepathy.com/blog/business/strategic-ux-the-art-of-reducing-friction

[20] D. Chaiffetz, "3 WAYS FRICTION CAN IMPROVE YOUR UX." [Online]. Available: http://blog.invisionapp.com/3-ways-friction-can-improve-your-ux/

[21] S. G. Millstein, H. Heger, and G. a. Hunter, "Prosthetic use in adult upper limb amputees: a comparison of the body powered and electrically powered prostheses." *Prosthetics and orthotics international*, vol. 10, no. 1, pp. 27–34, 1986. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/3725563

[22] L. Bouzereau, *Star Wars: The Annotated Screenplays: Star Wars–a New Hope, The Empire Strikes Back, Return of the Jedi.* Lucas Books, 1997.

[23] L. Resnik, S. L. Klinger, and K. Etter, "The DEKA Arm: its features, functionality, and evolution during the Veterans Affairs Study to optimize the

DEKA Arm." *Prosthetics and orthotics international*, vol. 38, no. 6, pp. 492–504, 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24150930

[24] E. Guizzo, "Dean Kamen's "Luke Arm" Prosthesis Receives FDA Approval." [Online]. Available: http://spectrum.ieee.org/automaton/biomedical/bionics/dean-kamen-luke-arm-prosthesis-receives-fda-approval

[25] CBS, "Bionic DEKA Arm, mind-controlled prosthetic, approved by FDA." [Online]. Available: http://www.cbsnews.com/news/bionic-deka-arm-mind-controlled-prosthetic-approved-by-fda/

[26] W. Selpho, "William selpho," 1857.

[27] A. Hess, "Body-Powered Prostheses." [Online]. Available: http://www.upperlimbprosthetics.info/index.php?p=1_9_Body-Powered

[28] B. Radocy, "Special Considerations: Upper-Limb Prosthetic Adaptations for Sports and Recreation," in *Atlas of Limb Prosthetics: Surgical, Prosthetic, and Rehabilitation Principles*, 2nd ed., H. Bowker and J. Michael, Eds. American Academy of Orthopedic Surgeons, 1992, ch. 12C Specia. [Online]. Available: http://www.oandplibrary.org/alp/chap12-03.asp

[29] A. Hess, "Adaptive Prosthetics." [Online]. Available: http://www.upperlimbprosthetics.info/index.php?p=1_24_Adaptive

[30] S. McNutt, "Adaptive Gardening." [Online]. Available: http://www.amputee-coalition.org/resources/adaptive-gardening/

[31] BeBionic, "BeBionic Hand." [Online]. Available: http://bebionic.com/the_hand

[32] OttoBock, "Otto Bock Michelangelo Prosthetic Hand." [Online]. Available: http://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/michelangelo-prosthetic-hand/

[33] TouchBionics2, "Touch Bionics Quantum." [Online]. Available: http://www.touchbionics.com/products/active-prostheses/i-limb%E2%84%A2-quantum

[34] TouchBionics3, "How the iLimb Works." [Online]. Available: http://www.touchbionics.com/products/how-i-limb-works

[35] VincentSystems, "Vincent Systems Evolution 2." [Online]. Available: http://vincentsystems.de/en/prosthetics/vincent-evolution-2/

[36] L. A. Miller, R. D. Lipschutz, K. A. Stubblefield, B. A. Lock, H. Huang, T. W. Williams, R. F. Weir, and T. A. Kuiken, "Control of a Six Degree of Freedom Prosthetic Arm After Targeted Muscle Reinnervation Surgery," *Archives of Physical Medicine and Rehabilitation*, vol. 89, no. 11, pp. 2057–2065, 2008.

[37] T. a. Kuiken, G. a. Dumanian, R. D. Lipshutz, L. a. Miller, and K. a. Stubblefield, "The use of targeted muscle reinnervation for improved myoelectric prosthesis control in a bilateral shoulder disarticulation amputee," *Prosthetics and Orthotics International*, vol. 28, pp. 245–253, 2004.

[38] 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 Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 1, pp. 37–45, 2008.

[39] T. A. Kuiken, "No Title." [Online]. Available: http://l-homme-bionique.e-monsite.com/pages/jour-7-post-operation.html

[40] A. E. Schultz and T. A. Kuiken, "Neural Interfaces for Control of Upper Limb Prostheses: The State of the Art and Future Possibilities," *PM and R*, vol. 3, no. 1, pp. 55–67, 2011.

[41] E. Biddiss and T. Chau, "Upper-limb prosthetics: critical factors in device abandonment." *American journal of physical medicine & rehabilitation /*

Association of Academic Physiatrists, vol. 86, no. 12, pp. 977–987, 2007. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18090439

[42] O. Yetkin, R. Karulkar, S. K. Das, and D. O. Popa, "Control of a Powered Prosthetic Hand Via a Tracked Glove1," *Journal of medical devices*, vol. 9, no. 2, pp. 20 920–20 922, 2015.

[43] J. Kramer and L. Leifer, "The Talking Glove," *SIGCAPH Comput. Phys. Handicap.*, no. 39, pp. 12–16, 1988. [Online]. Available: http://doi.acm.org/10.1145/47937.47938

[44] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *Computer Graphics and Applications, IEEE*, vol. 14, no. 1, pp. 30–39, 1994.

[45] C. Systems, "Cybergrasp Device." [Online]. Available: http://www.cyberglovesystems.com/cybergrasp/

[46] J. LaViola and R. Zeleznik, "Flex and Pinch: A Case Study of Whole-Hand Input Design for Virtual Environment Interaction," *International Conference on Computer Graphics and Imaging'99*, pp. 221–225, 1999.

[47] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo, *Time-of-Flight Cameras and Microsoft KinectTM*. Springer Science & Business Media, 2012.

[48] G. Marin, F. Dominio, and P. Zanuttigh, "HAND GESTURE RECOGNITION WITH LEAP MOTION AND KINECT DEVICES Giulio Marin , Fabio Dominio , Pietro Zanuttigh Department of Information Engineering , University of Padova," *International Conference on Image Processing(ICIP)*, pp. 1565–1569, 2014.

[49] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier, "Digits: Freehand 3D Interactions Anywhere Using a Wrist-Worn Gloveless Sensor," *Proceedings of the 25th annual ACM symposium on User*

interface software and technology - UIST '12, p. 167, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2380116.2380139

[50] R. Nuwer, "Armband adds a twitch to gesture control," *New Scientist*, vol. 217, no. 2906, p. 21, 2013.

[51] T. TORRES, "Control Your PC With Arm GesturesSort Of." p. 60, 2015. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&db=f5h&AN=103333135&site=eds-live

[52] Z. Abraham, T. Solomon, M. Xie, and K. Yeh, "Control of an Affordable Hand and Wrist Prosthesis," 2015.

[53] A. Heinrich, "Gest glove has gesture control on hand," *Gizmag*, nov 2015. [Online]. Available: http://www.gizmag.com/gest-gesture-controller-glove/40174/

[54] Y. Sun, J. M. Hollerbach, and S. A. Mascaro, "Predicting fingertip forces by imaging coloration changes in the fingernail and surrounding skin," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 10, pp. 2363–2371, 2008.

[55] L. Hode and K. H. Norris, "Penetration of light into living tissue." [Online]. Available: http://www.laser.nu/lllt/pdf/Penetration.pdf

[56] T. Vo-Dinh, "Basic Instrumentation in Photonics," in *Biomedical Photonics Handbook*, 2003. [Online]. Available: http://www.crcnetbase.com/doi/book/10.1201/9780203008997

[57] M. S. Patterson, B. Chance, and B. C. Wilson, "Time resolved reflectance and transmittance for the non-invasive measurement of tissue optical properties," *Applied Optics*, vol. 28, no. 12, pp. 2331–2336, 1989.

[58] J. Hardesty and B. Attili, "Spectrophotometry and the Beer-lambert Law: An Important Analytival technique in Chemistry." *Department of Chemistry, Collin College*, pp. 1–6, 2010.

[59] W.-F. Cheong, S. A. Prahl, and A. J. Welch, "A Review of the Optical Properties of Biological Tissues," *Ieee Journal of Quantum Electronics*, vol. 26, no. 12, pp. 2166–2185, 1990.

[60] JohnsHopkins, "Johns Hopkins Health Library – Electromyography (EMG)." [Online]. Available: http://www.hopkinsmedicine.org/healthlibrary/test_procedures/neurological/electromyography_e

[61] S. A. Dalley, H. A. Varol, and M. Goldfarb, "Multigrasp myoelectric control for a transradial prosthesis," *IEEE International Conference on Rehabilitation Robotics*, 2011.

[62] B. Karlk, "Machine Learning Algorithms for Characterization of EMG Signals," *International Journal of Information and Electronics Engineering*, vol. 4, no. 3, pp. 189–194, 2014. [Online]. Available: http://www.ijiee.org/index.php?m=content&c=index&a=show&catid=45&id=465

[63] A. Soares, A. Andrade, E. Lamounier, and R. Carrijo, "The development of a virtual myoelectric prosthesis controlled by an EMG pattern recognition system based on neural networks," *Journal of Intelligent Information Systems*, vol. 21, no. 2, pp. 127–141, 2003.

[64] S. S. Nair, R. M. French, D. Laroche, and E. Thomas, "The application of machine learning algorithms to the analysis of electromyographic patterns from arthritic patients," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 2, pp. 174–184, 2010.

[65] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Transactions on Bio-*

Medical Engineering, vol. 50, no. 7, pp. 848–54, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1206493\nhttp://www.ncbi.nlm.nih.gov/pu

[66] G. N. Saridis and T. P. Gootee, "EMG pattern analysis and classification for a prosthetic arm." *IEEE transactions on bio-medical engineering*, vol. 29, no. 6, pp. 403–412, 1982.

[67] P. Parker, K. Englehart, and B. Hudgins, "Myoelectric signal processing for control of powered limb prostheses," *Journal of Electromyography and Kinesiology*, vol. 16, no. 6, pp. 541–548, 2006.

[68] Y. Huang, K. B. Englehart, B. Hudgins, and A. D. C. Chan, "A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 11, pp. 1801–1811, 2005.

[69] A. D. C. Chan and K. B. Englehart, "Continuous myoelectric control for powered prostheses using hidden Markov models," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 1, pp. 121–124, 2005.

[70] P. Gallant, E. Morin, and L. Peppard, "Feature-based classification of myoelectric signals using artificial neural networks," *Medical and Biological Engineering and Computing*, vol. 36, no. 4, pp. 485–489, 1998. [Online]. Available: http://dx.doi.org/10.1007/BF02523219

[71] E. Scheme, A. Fougner, Stavdahl, A. D. C. Chan, and K. Englehart, "Examining the adverse effects of limb position on pattern recognition based myoelectric control," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, 2010, pp. 6337–6340.

[72] 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*, vol. 3, no. 2, pp. 175–180, 2008.

[73] J. L. G. Nielsen, S. Holmgaard, N. Jiang, K. B. Englehart, D. Farina, and P. A. Parker, "Simultaneous and proportional force estimation for multifunction myoelectric prostheses using mirrored bilateral training," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 3 PART 1, pp. 681–688, 2011.

[74] N. Jiang, J. L. Vest-Nielsen, S. Muceli, and D. Farina, "EMG-based simultaneous and proportional estimation of wrist/hand dynamics in uni-Lateral trans-radial amputees," *Journal of NeuroEngineering and Rehabilitation*, vol. 9, no. 1, p. 42, 2012.

[75] S. Muceli, S. Member, D. Farina, and S. Member, "Simultaneous and Proportional Estimation of Hand Kinematics From EMG During Mirrored Movements at Multiple Degrees-of-Freedom," vol. 20, no. 3, pp. 371–378, 2012.

[76] S. Maier and P. V. D. Smagt, "Surface EMG suffices to classify the motion of each finger independently," *... International Conference on Motion ...*, pp. 1–9, 2008. [Online]. Available: http://www.researchgate.net/publication/224999670_Surface_EMG_suffices_to_classify_the_motion

[77] M. Wininger, N.-H. Kim, and W. Craelius, "Pressure signature of forearm as predictor of grip force," *Journal of rehabilitation research and development*, vol. 45, no. 6, p. 883, 2008.

[78] D. Yungher and W. Craelius, "Discriminating 6 grasps using force myography of the forearm," in *BMES Annual Fall Meeting*, 2006.

[79] S. L. Phillips and W. Craelius, "Residual kinetic imaging: a versatile interface for prosthetic control," *Robotica*, vol. 23, no. 03, pp. 277–282, 2005.

[80] C. Cipriani, M. Controzzi, and M. C. Carrozza, "The SmartHand transradial prosthesis." *Journal of neuroengineering and rehabilitation*, vol. 8, no. 1, p. 29, 2011. [Online]. Available: http://www.jneuroengrehab.com/content/8/1/29

[81] E. a. Biddiss and T. T. Chau, "Upper limb prosthesis use and abandonment: a survey of the last 25 years." *Prosthetics and orthotics international*, vol. 31, no. 3, pp. 236–257, 2007.

[82] L. V. McFarland, S. L. Hubbard Winkler, A. W. Heinemann, M. Jones, and A. Esquenazi, "Unilateral upper-limb loss: satisfaction and prosthetic-device use in veterans and servicemembers from Vietnam and OIF/OEF conflicts." *Journal of rehabilitation research and development*, vol. 47, no. 4, pp. 299–316, 2010.

[83] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill, "A hand gesture interface device," *ACM SIGCHI Bulletin*, vol. 17, no. SI, pp. 189–192, 1986.

[84] M. Banzi, *Getting Started with Arduino (Make: Projects)*, 2008, vol. 11. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0596155514\nhttp://www.amazon.com/Getting-Started-Arduino-Massimo-Banzi/dp/1449309879

[85] F. Scholkmann, S. Spichtig, T. Muehlemann, and M. Wolf, "How to detect and reduce movement artifacts in near-infrared imaging using moving standard deviation and spline interpolation." *Physiological measurement*, vol. 31, no. 5, pp. 649–662, 2010.

[86] S. Jin, X. Ning, and G. A. Mirka, "An algorithm for defining the onset and cessation of the flexion-relaxation phenomenon in the low back musculature," *Journal of Electromyography and Kinesiology*, vol. 22, no. 3, pp. 376–382, 2012.

[87] L. Fraiwan, K. Lweesy, A. Al-Nemrawi, S. Addabass, and R. Saifan, "Voiceless Arabic vowels recognition using facial EMG," *Medical and Biological Engineering and Computing*, vol. 49, no. 7, pp. 811–818, 2011.

[88] S. Daniels, "Statistics on the Arduino," 2012. [Online]. Available: http://provideyourown.com/2012/statistics-on-the-arduino/

[89] V. Mathiowetz, S. Federman, and D. Wiemer, "Box and Block Test of Wan al Dex , Norms for 6-19 Year Olds," *Canadian Journal of Occupational Therapy*, vol. 52, no. 5, pp. 241–245, 1985.

[90] O. Yetkin, K. Wallace, J. Sanford, and D. O. Popa, "Control of a Powered Prosthetic Device via a Pinch Gesture Interface," *SPIE Sensors for Next-Generation Robotics II*, 2015.

[91] Encyclopedia.com, "Reaction Time," 1968. [Online]. Available: http://www.encyclopedia.com/doc/1G2-3045001044.html

[92] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier, "Digits: Freehand 3D Interactions Anywhere Using a Wrist-Worn Gloveless Sensor," *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, p. 167, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2380116.2380139

[93] GEST, "GEST Wearable Device." [Online]. Available: www.gest.co

[94] S. A. Mascaro and H. Harry Asada, "Photoplethysmograph fingernail sensors for measuring finger forces without haptic obstruction," pp. 698–708, 2001.

[95] G. Strangman, D. A. Boas, and J. P. Sutton, "Non-invasive neuroimaging using near-infrared light," pp. 679–693, 2002.

[96] J. A. Fishel, *Design and use of a biomimetic tactile microvibration sensor with human-like sensitivity and its application in texture discrimination using Bayesian exploration.* University of Southern California, 2012.

[97] W. E. Hick, "On the rate of gain of information," *Quarterly Journal of Experimental Psychology*, vol. 4, no. 1, pp. 11–26, 1952.

[98] F. Sebelius, L. Eriksson, C. Balkenius, and T. Laurell, "Myoelectric control of a computer animated hand: a new concept based on the combined use of a tree-structured artificial neural network and a data glove." *Journal of medical engineering & technology*, vol. 30, no. 1, pp. 2–10, 2006.

BIOGRAPHICAL STATEMENT

Oguz Yetkin was born in Izmir, Turkey in 1975. He received his B.S. degree from The University of Wisconsin-Madison in 1999.