2017

An integrated system for quantitatively characterizing different handgrips and identifying their cortical substrates

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Thesis

AN INTEGRATED SYSTEM FOR QUANTITATIVELY CHARACTERIZING DIFFERENT HANDGRIPS AND IDENTIFYING THEIR CORTICAL SUBSTRATES

by

MAHTAB ALAM

B.Eng., The City College of New York, 2015

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2017

Approved by

First Reader

Lucia M. Vaina, M.D., Ph.D.
Professor of Biomedical Engineering
Boston University, College of Engineering

Research Professor of Neurology
Boston University, School of Medicine

Second Reader

Lee N. Marinko, PT, Sc.D., OCS, OMT, FAAOMPT
Clinical Assistant Professor of Physical Therapy and Athletic Training
Boston University, Sargent College of Health and Rehabilitation Sciences

Third Reader

Sandor Vajda, Ph.D.
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Chemistry

DEDICATION

This thesis is dedicated to ammu, abbu, and Madiha.

-Redoy

ACKNOWLEDGMENTS

This Master's thesis is the culmination of one year's worth of work and could not have been completed without the efforts of all those who helped me along the way. First and foremost, I would like to thank my advisor, Professor Lucia M. Vaina, for providing me the opportunity to work on this exciting project and for guiding me in the process of carrying out this research. I also have to credit Lucia for her valuable encouragement, especially at the times when I thought I was not capable of pushing myself further. Many thanks to my committee members Lee N. Marinko and Sandor Vajda, for offering their insightful comments at various stages and their enthusiasm about this project. I am also thankful to Dimitrios Pantazis at MIT for helping with the MEG scanning and for his suggestions on the design of the behavioral test paradigm.

I am extremely gracious for the support from all the members of the Brain and Vision Research Laboratory (BraviLab) – both from a scientific and social standpoint. Thank you to Yansong Geng and Funda Yildirim for various help with my research and for making the lab environment a pleasant place to work. An extremely heartfelt thanks to Kunjan Rana, who was readily available to answer any of my technical questions, even while he was busy completing his own PhD thesis. I am grateful to the BME department at Boston University for the opportunity and resources to undertake this research.

I must also express my gratitude to my hard-working parents, Sayla Sultana and Mohammed Alam, and my younger sister, Madiha, for their unconditional love and support. I am also extremely fortunate to have a strong support network in some of my closest friends from back home in New York – a shout out to the Jamaica Crew: Zubair,

Sejal, Ryeen, Alvi, Rizwan, and Raihan; the City College guys: Ahmed, Amal, and Mohammad; and one of the few Stuy people I still talk to: Hossam – all of whom are accomplishing or on their way towards accomplishing great things in their respective fields as well. Lastly, I have to recognize the Legend of Zelda video game soundtrack and the coffee machine in the BME grad lounge for helping me power through some of the most challenging parts of this thesis.

AN INTEGRATED SYSTEM FOR QUANTITATIVELY CHARACTERIZING DIFFERENT HANDGRIPS AND IDENTIFYING THEIR CORTICAL SUBSTRATES

MAHTAB ALAM

Boston University College of Engineering, 2017

Major Professor: Lucia M. Vaina, Ph.D., M.D., Professor of Biomedical Engineering, College of Engineering, Research Professor of Neurology, School of Medicine

ABSTRACT

Motor recovery of hand function in stroke patients requires months of regular rehabilitation therapy, and is often not measured in a quantitative manner. The first goal of this project was to design a system that can quantitatively track hand movements and, in practice, related changes in hand movements over time. The second goal of this project was to acquire hand and finger movement data during functional imaging (in our case we used magnetoencephalography (MEG)) to be used for characterizing cortical plasticity associated with training. To achieve these goals, for each hand, finger flexion and extension were measured with a data glove and wrist rotation was calculated using an accelerometer. To accomplish the first goal of the project, we designed and implemented Matlab algorithms for the acquisition of behavioral data on different handgrips, specifically power and precision grips. We compiled a set of 52 objects (26 man-made and 26 natural), displayed one at the time on a computer screen, and the subject was asked to form the appropriate handgrip for picking up the object image presented. To accomplish the second goal, we used the setup described above during an MEG scanning

session. The timescales for the signals from the glove, accelerometer, and MEG were synchronized and the data analyzed using Brainstorm. We validated proper functionality of the system by demonstrating that the glove and accelerometer data during handgrip formation correspond to the appropriate neural responses.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

Proper hand function plays an essential role in the range of human movement, with the hand being one of most important and frequently used external organs of the human body. Hand and finger movement are needed for a variety of actions related to object manipulation, communication, and other activities of everyday living (ADL). Limitations in hand motor functions can lead to severe negative impacts on quality of life.

Stroke is one of the leading causes of long-term motor disability in the US (American Heart Association). Chronic motor deficits are common, particularly for the hand, with impairments such as inability to fully form grips around objects (Cruz et al., 2005; Kamper and Rymer, 2001) and delays in grip and release timing (Nowak et al., 2003; 2007). These hand impairments can continue to exist long after upper limb recovery. Even partial recovery of hand functions can require months of rehabilitation.

Assessment of motor recovery of hand function is needed for the planning of rehabilitation processes. These tests of function typically require the patient to carry out certain fundamental tasks or simple grips to measure the action time for each. However, evaluating the functional capabilities during hand recovery can be made more comprehensive. Many of the components that characterize functional hand use, including finger extension/flexion and hand displacement in space, are measureable with sensitive instrumentation. Rehabilitation should be targeted to each patient's particular deficits.

For the first part of this project, we propose to design and implement a system that can reliably record quantitative information about hand movements. Characterizing

different handgrips by measuring finger and wrist movements can be used for experiments that look at changes or improvements in these measurements over time, during rehabilitation. In the second part of the project, we propose combining the hand quantification system with a functional neuroimaging technique, in order to look at neural activity during handgrip activity. For hand function recovery, this integrated system can be used to study cortical plasticity associated with training over time.

Both parts of the project require designing a behavioral test to administer to subjects, in order to record handgrip responses. The proposed test will require subjects to form power or precision grips in response to images of everyday objects presented on a computer screen. For the first objective, this will produce a dataset of hand movement information. For the second objective, it will provide hand movement data as well as the corresponding neural signals based on the stimuli and responses. We hypothesize that we will be able to identify the specific areas of activation through magnetoencephalography (MEG) imaging. We plan to demonstrate that the data obtained from the handgrip quantification can be correlated with the data from the MEG scanning.

Part 1: Quantitative tracking of hand movements

The first part of this project required interfacing the hardware used for hand movement tracking with Matlab and running a test to collect meaningful information about the function of the system. Although handgrips being formed during a behavioral test can be observed qualitatively, we aim to further describe the hand activity. Such an approach provides more detailed information about the particular components of the hand movements from the subject. While many studies have developed devices to measure one

specific hand motion or focus on individual fingers, we aim to characterize the whole hand movement in order to gather multiple metrics simultaneously.

The particular hardware selected for measuring hand movements is a pair of MRI and MEG compatible data gloves along with a pair of 3-axis accelerometers. The gloves can collect information about the flexion and extension movements of each finger as well adduction and abduction from between each finger. The accelerometers measure the acceleration of motion in the x, y, and z directions. The placement of the accelerometers on the back of the hand provides 3D orientation data about the rotational motion of the wrist. The raw acceleration vectors can be analyzed to determine the angle of rotation. It is important to note that no forces are measured with these components. Only the changes in position over time are measured. Since our imaging functions and tests are already prepared in the Matlab software environment, we developed the integrated system in Matlab as well. Once the gloves were made compatible with Matlab, separate scripts were written to record and save data, specifying sampling rates and setting timestamps. Matlab also simplifies the data collection and data analysis steps.

Part 2: Acquire movement data during imaging

In the second part of this project, the setup from the first section was run during functional neuroimaging. The most important purpose of developing such a system is to draw useful insights, correlating hand movement activity with neural imaging data. In an increasingly data-driven research environment, the ability to quantify every component in the data collection process is valuable for making novel discoveries. This unified system can serve as a tool to facilitate that.

MEG was selected as the neuroimaging modality to be used for this project due to its high temporal resolution and ability to study functional connectivity. The MEG scanner has its own software that collects and saves the MEG data. The accelerometers are also connected to the MEG machine as an additional input source and their data is saved along with the MEG signals. Our Matlab scripts can incorporate certain time markers and stimulus signal markers into this data to aid with post-processing. Additionally, subjects perform a behavioral test during scanning, which is developed within Matlab using BraviShell, a Psychtoolbox-based Matlab library used by our lab to develop behavioral and psychophysical tests for neuroimaging studies. A single script will finally be produced to collect data from the glove as well as run the behavioral task in a manner that corresponds with the MEG data collection.

The overall purpose of this project is to develop a unified system for simultaneously collecting data about the subject's hand movements as well as neural imaging. Such a system would be particularly useful for neuroimaging experiments studying the motor function of the hand. The first objective is to demonstrate proper functioning of the hand movement tracking system by collecting recordings from healthy subjects. The second objective is to validate that reliable signals can be recorded from the hand movement tracking system during a functional neuroimaging session and that the hand movement data corresponds with the neural signals.

CHAPTER TWO: BACKGROUND

2.1 Hand movement tracking devices

The accelerometers being used are a pair of MEG and MRI compatible three-axis accelerometers (ADXL330 iMEMS Accelerometer, Analog Devices, Inc., Norwood, MA, USA). The accelerometer is connected to a separate unit that serves to filter the incoming signals as well as house the battery, switch, and one BNC connector output for each of the three axes. The accelerometers and attached unit were developed and provided by Veikko Jousmäki (Aalto University, Finland). It generates three voltage values between 0 and 5 V, representing the acceleration in the x, y, and z axes.

It is important to make the distinction between acceleration, velocity, and displacement, as the accelerometers do not directly measure change in position. The magnitude of the output increases or decreases as a function of acceleration. For example, translation of the accelerometer along a single axis would show a steady rise followed by a steady fall, with the magnitude depending on how "sudden" the motion is. At rest, with the z-axis aligned to the Earth's gravitational field, the z-axis output will be slightly offset from the other two axes, representing the acceleration due to gravity. This constant acceleration from gravity detected by the accelerometer will be used later for calculating the wrist rotation angle from the raw accelerometer data. The maximum range of acceleration measured is ±3g, or three times the acceleration of gravity.

This accelerometer has been used in a series of studies by the lab that developed it, to explore the concept of corticokinematic coherence (CKC), or the synchronization between MEG oscillations and voluntary movement kinematics. The hand and finger

kinematics were measured during MEG using this accelerometer. In these papers, they quantified the coupling between MEG signals and repetitive voluntary hand movements (Bourguignon et al., 2012), determined the contribution of active and passive finger movements to CKC (Piitulainen et al., 2013), and searched for the brain areas that display coherent MEG activity with observed hand movements (Bourguignon et al., 2013). The successful implementation of the accelerometers in MEG studies validates their use in further research.

The data gloves used for collecting handgrip data are the 5DT Ultra 14 Data Gloves (Fifth Dimension Technologies, Irvine, CA, USA). Like the accelerometers, these data gloves are also MEG and MRI compatible. They measure individual finger flexion as well as abduction between the fingers by the use of 14 fiber optic cable sensors along the back of the hand. (It should be noted that the gloves do not measure force.) Ten flexion sensors are at two joints in each finger and four abduction sensors are in between each finger. The sensors on the glove correspond to the metacarpophalangeal joints (MCP), proximal interphalangeal joints (PIP), and distal interphalangeal joints (DIP). The fiber optic sensors are connected to a driver, which contains the software, and interfaces with the computer via USB connection. The output value for each sensor is a scaled measurement between 0 and 1 (12-bit resolution), with 1 representing full flexion.

Although the 5DT Ultra 14 Data Gloves are a commercial product used primarily for animation and virtual reality applications, they are also suitable for a wide range of research applications. For example, the gloves have been used for administering remotely-monitored exercises for cerebral palsy patients (Huber et al., 2010), developing

an online signature verification system (Kamel et al., 2008), and monitoring gestures of the elderly and patients for implicit communication (Rajendran et al., 2013). There is also literature relating to brain machine interfaces, such as using the gloves to quantify hand movements for classifying electrocorticography signals (Chestek et al., 2013), tracking individual finger movements to determine coherence with electrocorticography signals (Wang et al., 2009), and controlling a cursor with the gloves while deep brain stimulation is applied (Hanson et al., 2012).

2.2 Handgrips

In his 1956 paper, J.R. Napier was the first to categorize handgrips into power and precision grips (Napier, 1956). Virtually the entire range of human handgrip activity can be characterized as one of the two, based on anatomical and functional capabilities. Such a formally defined system is valuable for research and rehabilitation purposes. In the power grip, the object is held with partially flexed fingers and the palm. It is characterized by flexing all the fingers against the palm to grasp the object in a very steady grip. The precision grip involves pinching a small object between the thumb and one or more of the flexor aspects of the fingers (Napier, 1956) and usually between the thumb and the index finger (Vainio et al., 2007).

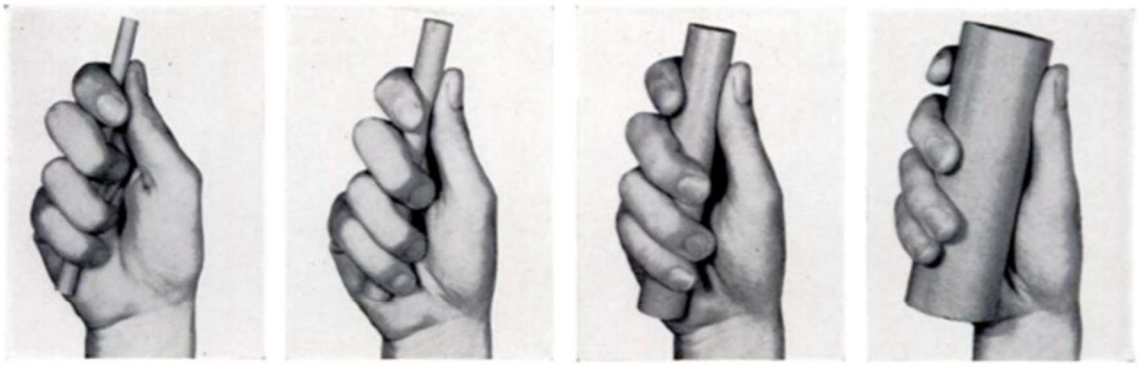

FIG. 8
An arbitrarily chosen series of postures illustrating some of the phases of the power grip complex.

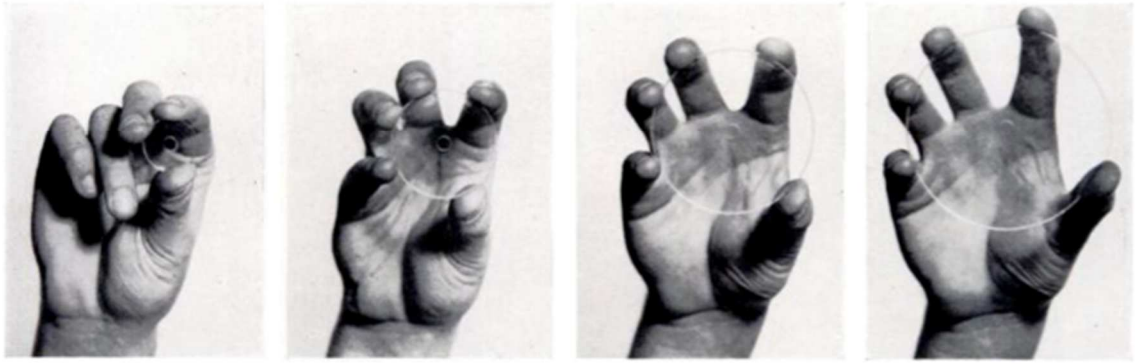

FIG. 9
An arbitrarily chosen series of postures illustrating some of the phases of the precision grip complex.

Figure 2-1: Various examples of power and precision grips. (Napier, 1956)

Each of these grips has a different neural circuitry. In human neuroimaging studies, it has been shown that M1 is utilized during a variety of voluntary hand movements (Roland and Zilles, 1996). Nonhuman primate studies have found that certain M1 neurons that directly influence the hand muscles are active during precision but not during power grip despite the target muscles being active in both grips (Muir and Lemon, 1983). In an fMRI study of precision and power grip tasks (Ehrsson et al., 2000), it was found that activity was more pronounced in the contralateral primary sensory and primary motor cortex for power grip than for precision grip. However, during the precision grip

task, there was more activation in the ipsilateral ventral premotor area, rostral cingulate motor area, and locations in the posterior parietal and prefrontal cortices. Other studies have shown asymmetric representations of the two grips in the left and right hemispheres, with the left hemisphere playing a special role in the visual control of precision grip movements (Gonzalez et al., 2006).

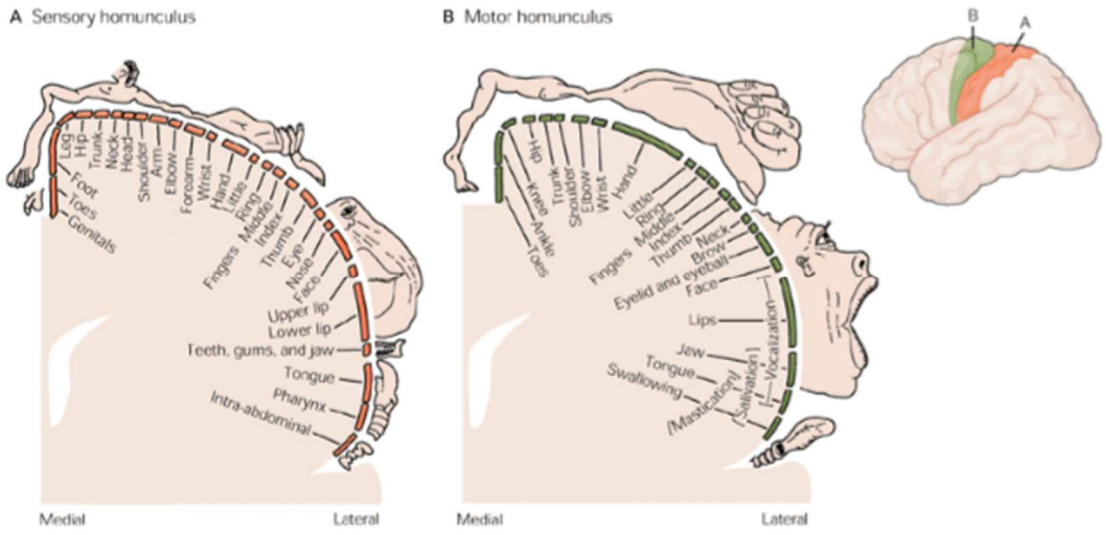

Figure 2-2: SI and MI brain areas. The motor representation of the hand is significant. (Kandel et al., 2000)

There are various factors influencing whether an object is grasped using a power or precision grip. Visual features of the object, such as shape and size, help guide the posture of the hand. It has been demonstrated that performing a power or precision grip is correlated with the size of the object presented (Grezes et al., 2003). In the study, subjects were asked to form a power or precision grip while viewing small objects (which would presumably be grasped with a precision grip) and large objects (presumably grasped with a power grip). Different combinations were presented, with some trials including objects

matching grip command (small object viewed with precision grip command or large object viewed with power grip command) and other trials with opposite stimuli (large object viewed with precision grip or small object viewed with power grip). The reaction time for the non-matching trials was longer, indicating that viewing an object corresponding with the grip being formed led to a quicker response.

Another object feature associated with the individual's decision about how to manipulate it is whether the object is natural or man-made. There are important distinctions made between these two categories. Examples of natural objects include a peanut, banana, flower, or leaf. Man-made objects include a key, pen, bottle, or hammer. Evidence suggests that functional features define man-made object recognition while visual features are initially noted in natural type objects (Borghi et al., 2007). Vision and action have been shown to have connections previously.

2.3 Stroke recovery

While there are many causes for stroke, a general definition is that a stroke occurs when a certain part of the brain is deprived of blood flow due to a blood clot or ruptured vessel (NIH). This leads to immediate symptoms of sudden weakness, vision loss, or numbness of the face, arms, or legs. Long-term effects include loss of motor function, speech, or memory. Stroke patients often lose proper function of upper extremities, including the hands and fingers. Rehabilitation exercises attempt to help restore ability to carry out everyday tasks and improve quality of life (NIH). Upper extremity recovery, particularly of the hands, is crucial for carrying out daily activities. Methods of assessing a patient's hand and finger mobility include the Fugl-Meyer Assessment for Upper

Extremities and the Jebson Hand Function Test. Neuroimaging studies associated with recovery after stroke have been used for neurofeedback treatment (Linden et al., 2016) and to study functional connectivity and neural reorganization (Westlake et al., 2011). Similarly, in this thesis, we aim to develop a system that will ultimately use neuroimaging, along with hand motor function tracking, to study functional connectivity and plasticity changes during the course of hand motor function recovery from stroke.

2.4 MEG scanning

The functional neuroimaging modality selected for this project is magnetoencephalography (MEG). MEG scanning is a non-invasive neuroimaging technique that records neuronal magnetic activity over time, and has proven useful for research in cognitive brain processes and determining the function of brain regions. One of the primary advantages of MEG over other functional neuroimaging techniques such as fMRI is the high temporal resolution, on the order of milliseconds (Hämäläinen et al., 1993). Additionally, MEG measures brain activity with a higher number of sensors than a modality such as electroencephalography (EEG), offering better spatial resolution. Therefore, MEG is optimal for investigating time-sensitive cortical activity of continuous neurophysiological processes. However, MEG is only sensitive to cortical activity, as it is practically unable to detect deep brain activity due to the rapid decay of signal at further distances from the source (Hämäläinen et al., 1993). It also does not identify current flow perpendicular to the surface of the skull because of the approximate spherical shape of the skull canceling out radial primary currents and volume conduction currents.

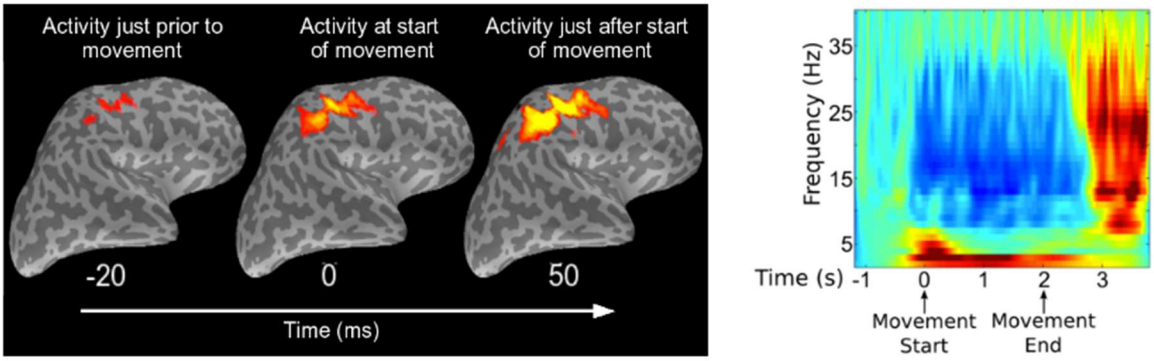

Figure 2-3: Example of MEG results. The activation map (left) shows the cortical surface at different time points (-20, 0, and 50 ms) with an increase in neural activity at each successive stage. MEG is convenient for distinguishing continuous neural processes at the millisecond scale, as demonstrated here. The frequency analysis (right) shows the neural oscillations in frequency bands (y-axis) that are most common, as denoted by the heatmap, over a selected time period (x-axis) during which the subject is performing some movement. In this example, delta waves (1–4Hz) are most common during the movement period and beta waves (13–30Hz) immediately afterwards. (martinos.org)

CHAPTER THREE: QUANTITATIVE TRACKING OF HAND MOVEMENTS

3.1 Introduction

In the first part of this project, we aim to demonstrate that hand movement data can be properly recorded using the hardware we've chosen. Code was written to interface both the gloves and accelerometer with Matlab, and the steps required to do so are described. An overview of the behavioral test is explained as well. Once fully prepared, the test was run on 20 healthy subjects. The data collected was analyzed and subjects' wrist motions and handgrips described.

The three components in the integrated system are the accelerometers, the data gloves, and the behavioral test in Matlab. The entire data acquisition and analysis process was run in Matlab, in order to maintain the same environment as the software that already exists and to facilitate the data analysis phase. Figure 3-1 provides an overview of the functionality of each component and the steps required to interface them and acquire data. The data collection code for each component was run simultaneously, parallel to one another, and the output data saved to one unified workspace in Matlab. (The data was saved externally as well, as .mat files and .xls files.) Analysis of the results was also carried out in Matlab.

Separate analyses were done on the glove and accelerometer data for each subject, and results were organized into time series per trial. The conditions compared were power vs precision grips and pronated vs neutral hand positions. Hand movements were also categorized using clustering algorithms. Further analyses were done to compare hand movement measurements across subjects.

3.2 Methods and Materials

The process of developing the hand movement tracking system consisted of selecting the individual components, evaluating their compatibility with each other, programming the ability to communicate, and incorporating robust data collection methods. Additionally, a behavioral test was developed for prompting handgrip formation in subjects. The purpose of the test was to present on-screen stimuli to subjects and obtain a behavioral response. Behavioral data was collected from 20 subjects.

A flowchart illustrating the entirety of the programming necessary for this part of the project is shown in Figure 3-1. The blue boxes represent the external tools needed to implement into the system, the yellow boxes are C/C++ code, and the orange boxes are processes that were carried out in Matlab. A red outline indicates that the components that were developed specifically for this project. As described in the figure, the programming can be broken down into three sections – the data gloves, the accelerometer, BraviShell. This chapter will explain the specific capabilities and the necessary steps for each.

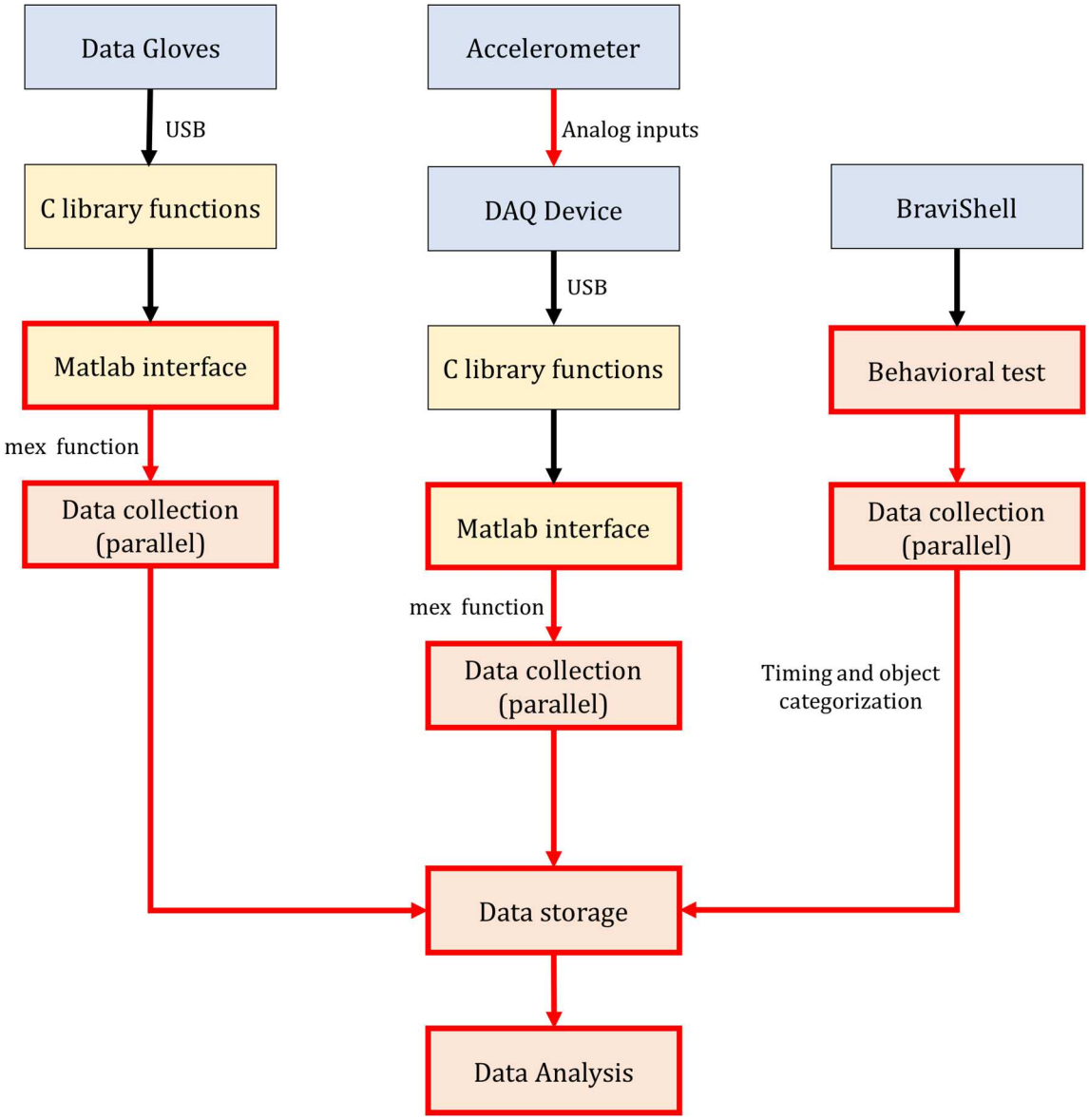

Figure 3-1: Flowchart illustrating parts of the code written for developing the hand movement tracking system. The blue boxes represent the separate external components of this system, the yellow boxes represent C/C++ code including the hardware SDK and wrapper code, and the orange boxes represent Matlab functions and scripts for reading, recording, and saving data. Arrows and boxes outlined in red were developed from scratch for this system. Descriptions adjacent to arrows describe the method used or information being transmitted at those steps.

3.2.1 Accelerometers

As described previously, the three-axis accelerometers used for the system are MRI and MEG compatible, and thus were appropriate for the second part of this project. Two accelerometers were set up for the system, but only one was used in this test, since we only studied the right hand. For this first part of the project, the accelerometers were interfaced directly with Matlab for data acquisition and analysis. However, for the second part of the project, the MEG machine includes several channels that support BNC inputs. During MEG scanning in the next part, the three accelerometer outputs were connected to the MEG machine and the accelerometer data was recorded in three separate channels alongside the MEG data. The maximum range of acceleration measured is ±3g, or three times the acceleration of gravity. Any movements during the behavioral test were well within this range.

To interface the accelerometer with Matlab, an intermediate step was necessary. A USB data acquisition device (DAQ) was used to collect data from the accelerometers. This DAQ converts analog signals to digital signals and connects to the computer via USB. We purchased a LabJack U6 USB DAQ, along with a LabJack CB37 Terminal Board, which attaches to the DAQ. The DAQ by itself includes only four analog inputs and the terminal board provides 14 additional analog inputs. For two accelerometers, a total of six analog inputs would be needed. The input range for the DAQ is ±10 V, which is sufficient for the 0 to 5 V accelerometer output. The DAQ uses screw terminal inputs for each channel, requiring wires to be connected from the three BNC outputs on each accelerometer unit (using an adapter) to a screw terminal for each analog input. A single

line for ground was also connected between the accelerometer and DAQ. Figure 3-2 is an image of what the setup looks like.

Figure 3-2: The three-axis accelerometer connected to the LabJack U6 USB DAQ device. The ADXL330 accelerometer (in black, on left) is connected to a unit (labeled "3-Axis 3G Accelerometer") that filters signals to three separate BNC connector outputs. Each of the BNC outputs (X, Y, Z) has an adapter and single wire connected to one of the LabJack U6 DAQ (labeled, in red) analog inputs (AIN0, AIN1, AIN2). There is also one ground connection from the BNC output (X) to the DAQ ground input (GND). The DAQ connects to a computer via USB.

Some initial testing of the accelerometer's signals was done using an oscilloscope. BNC connector cables were connected directly between the accelerometer and oscilloscope. The accelerometer was moved along a single axis at a constant frequency, and the oscilloscope readings observed. The output voltage range and axes directions were confirmed. The reliability of the DAQ was also tested using a function generator. Waveforms were input to the DAQ and the results observed on Matlab. Finally, a BNC

adapter from each accelerometer output was connected to the screw terminal inputs on the DAQ to view accelerometer readings on the computer.

The programming for this step was solely for interfacing the DAQ with Matlab. The accelerometers had no direct interaction with the code. The first step was to obtain readings from the DAQ on Matlab. The LabJack U6 DAQ manufacturer's software development kit (SDK) did not provide compatibility with Matlab. The solution to this was to develop a C wrapper code. This single script would be based on the existing C functions and we would rewrite them to return the desired values in Matlab. The DAQ drivers and SDK still needed to be installed since the wrapper calls on those libraries. The wrapper code defined the functions that would be relevant to this project. Specifically, the ability to initialize the DAQ, obtain readings from the DAQ, and close the DAQ. The wrapper code captures data from six analog input channels on the DAQ at a single point in time. To directly call the functions from the wrapper code in Matlab, it was compiled in Matlab using the mex function.

Wrapper command	**Matlab function**	**Purpose**
DAQinterfacer(1)	U6Open	Initializes the DAQ. Required for reading data.
DAQinterfacer(2)	U6Read	Captures data at a single point in time from the first six analog inputs on the DAQ.
DAQinterfacer(3)	U6Close	Disconnects the DAQ.

Table 3-1: List of functions written for interfacing the DAQ with Matlab.

After being compiled, separate Matlab functions were created for the open, read, and close commands. Since the read function only returns data from a single point in time, a separate Matlab script was written for recording data. This script defined the specifications for continuous recording, such as sampling rate, duration of recording, and

timestamps in the output data. The Matlab script saves the raw data directly to the Matlab workspace as well as externally to an .xls file. The two aspects taken into account for proper functionality with the remaining components in the system were timing and parallel processing. To establish a consistent timescale across the different data sources, the computer's internal clock was used to record the time, rather than a clock relative to the start of the function. Parallel processing was necessary to be able to run all the components in the system simultaneously, and the batch function in Matlab was used to do so.

3.2.2 Data gloves

As mentioned previously, the 5DT Ultra 14 Data Gloves, a pair of MEG-compatible optical sensor gloves were selected for this project. As with the accelerometers, both gloves were set up but only the right hand was used for this test. The output value between 0 and 1 is a scaled measurement, established every time the glove is initialized. The scaling is done automatically by the glove driver by setting 0 and 1 based on the maximum and minimum finger flexion of the user for that individual session. For this reason, our test always provided a calibration period before beginning to allow the user to set the maximum and minimum by extending and flexing all fingers as much as possible. Although not recorded for this test, the glove does also provide the capability to return raw uncalibrated data and this feature is included in the code developed for the project.

Figure 3-3. The 5DT Ultra 14 Data Gloves (right hand only pictured on left) has 14 optical sensors that are wired to a driver (black box on left). The driver connects to the computer via USB (not pictured). Each of the 14 optical sensors on the glove (right) are located at two joints on each finger and between each finger, along the back of the hand. (5dt.com)

The first step towards developing the integrated system was interfacing the data gloves with Matlab. As was the case with the DAQ, an SDK was provided with the gloves but did not include the capability to program directly in Matlab. The SDK functions were in C++ and a wrapper script was written that could be called from the Matlab environment to run the designated functions. The wrapper included the ability to initialize the gloves, identifying handedness, read calibrated data, read raw data, specify hand gesture (based on glove driver), and close the gloves. Once the wrapper code was compiled in Matlab, all these functionalities could be directly called upon in Matlab. In developing the wrapper, some key features had to be included. For example, since the gloves use a USB connection, we incorporated a method of detecting the USB port that the glove is connected to, rather than specifying the particular port each time the glove was called on. Additionally, we implemented the ability to connect both right and left gloves and differentiate handedness regardless of the order they were connected. Furthermore, for reading sensor data, we had to ensure that all 14 sensors were recorded at the same point in time.

Wrapper command	Matlab function	Purpose
gloveInterfacer2(0)	gloveOpen	Access USB ports to initialize the gloves. Required for reading data.
gloveInterfacer2(1)	gloveRead	Capture the scaled glove data at single point in time from all sensors in both gloves.
gloveInterfacer2(2)	gloveClose	Disconnect the gloves.
gloveInterfacer2(3)	gloveID	Determine which glove is right (denoted by 1) and left (denoted by 0).
gloveInterfacer2(4)	gloveReadRaw	Capture the raw glove data at a single point in time from all sensors in both gloves.
gloveInterfacer2(5)	gloveGesture	Output a number indicating the gesture formed at that instant, as defined in the manual.

Table 3-2: List of functions written for interfacing the gloves with Matlab.

A single Matlab script was written to acquire and save all the glove data, since the C++ wrapper simply returns one matrix containing the sensor data at a single point in time. It also separates the glove data into right and left, sets the sampling rate, sets the duration of recording, removes redundancies, and saves externally to an .xls file. Just like the DAQ Matlab script, this one also included timing information and parallel processing capabilities. The script also included the ability to remove redundancies from the signals because data points were repeating every few samples. This was due to Matlab was sampling data points at a higher rate than the gloves were returning. Since the Matlab sampling rate was user-defined, it was possible to adjust it to approximately match the rate at which data points were being output from the glove driver. Consecutively repeating data points were removed by the script as well. Due to the computer's internal computational times, this did not give a uniform sampling frequency. However, during the analysis phase afterwards, the final output data was resampled to a constant rate in order to facilitate accurate data analysis. The data were saved to the Matlab workspace and to an external .xls file.

3.2.3 Subjects

20 healthy volunteers, college students (5 males, 15 females, age range 18–20 years, mean = 18.81, SD = 0.51) participated in this behavioral-only test. All participants were right-handed and none had a history of neurological or psychiatric disorders. 12 subjects had two blocks during their test session and 8 subjects had three blocks. Glove data was recorded from all subjects, but valid accelerometer data was only collected from 15 subjects. For the second part of the thesis, one healthy right-handed volunteer (male, age 29) with no known neurological or psychiatric disorders participated in the MEG scan. Informed consent according to the Declaration of Helsinki (2008) was obtained from each subject.

3.2.4 Behavioral test

The gloves and accelerometer, once fully interfaced, were able to collect data about the subject's hand movements. In addition to those components, however, a behavioral test containing visual stimuli was employed to elicit a handgrip response from the subject. This test (called "Image Presentation" or "IP") aims to collect data from subjects about the formation of different handgrips based on images of objects. Subjects were presented with a series of images of everyday objects on the screen and instructed to form the grip they would make to normally grip the object. Handgrip data was collected from the gloves and wrist rotation data from the accelerometer simultaneously during the IP test.

The test was designed in BraviShell, a Matlab software package developed in the Brain and Vision Laboratory based on the Psychophysics Toolbox extensions. BraviShell

facilitates the design and data recording for behavioral and psychophysical tests. It defines the test paradigm, processes any key presses, and exports the test information. For our purposes, the particular information that was most important from the test were the timing and object categorization for each trial.

The concept of the IP test paradigm is that the subject is presented with an image of an object and will respond with the handgrip they associate with grasping the object. When developing the test, numerous factors were taken into consideration. In addition to the picture of the object, the part of the object that we intend for the subject to hold is pointed out with an arrow, to reduce any ambiguity in the type of grip to be formed. As described in an earlier chapter, the two main grip types being studied in the test are power and precision grip. A power grip is when the object is held "in a clamp formed by the partly flexed fingers and the palm, counter pressure being applied by the thumb lying more or less in the plane of the palm" (Napier, 1956) i.e. the fingers and thumb wrap around the object and hold it against the palm of the hand. In a precision grip, the object is "pinched between the flexor aspects of the fingers and the opposing thumb" (Napier, 1956) i.e. the forefingers and thumb hold the object.

The other condition studied was the orientation of the object. Subjects were instructed to rotate their wrist to be able to form the grip around the object in the direction the image is presented and the position the arrow is designating. Each object was displayed in such an orientation that the subject would have to make a grip by rotating their hand in either a pronated (horizontal) or neutral (vertical) position. The resting starting position that subjects were instructed to make was with the hand lying flat

on the surface of the table. This “horizontal” orientation is when the hand is fully in pronation. The “vertical” orientation is when the hand is in a neutral position. For images that required a pronation response, the subject did not have to make much of a wrist rotation since their hand is already in that orientation. For the images that required a neutral response, subjects would rotate their wrist approximately 90 degrees clockwise.

The object images presented as visual stimuli are categorized into four groups – natural power grip, man-made power grip, natural precision grip, and man-made precision grip. As illustrated in Figure 3-4, each of these categories has 13 objects, for a total of 52 distinct objects. These 52 images are doubled in order to account for horizontal and vertical orientations, for a total of 104 visual stimuli. The images were all centered over a 640x640 pixel white background. In the test, this was centered on the screen with a gray background. The objects were not to scale with one another. However, objects that are smaller were presented smaller on the screen. Many of the objects selected were based on previous literature and there is an equal number for each group. The images were presented to subjects in a random order.

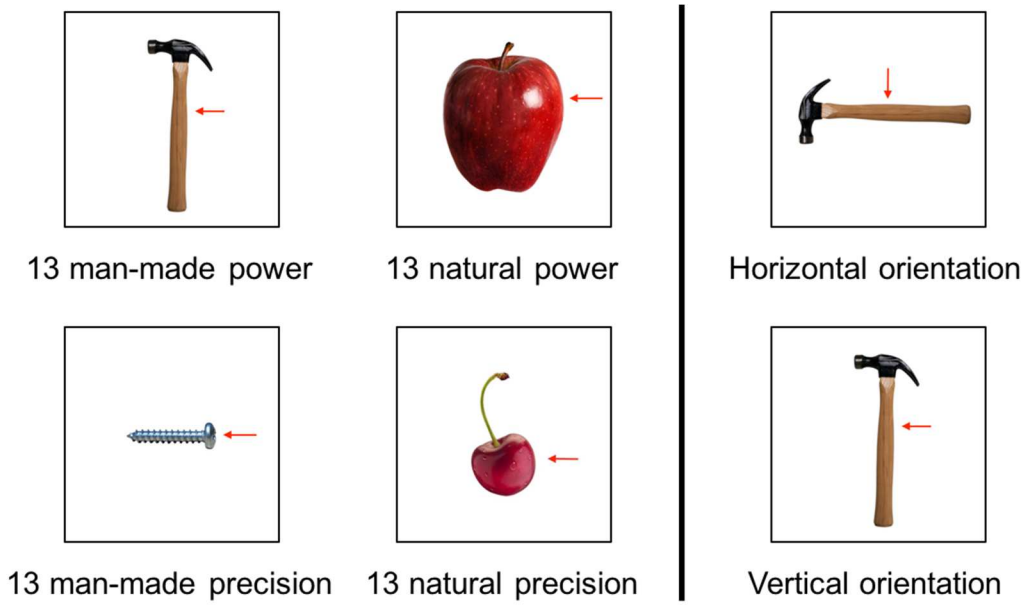

Figure 3-4: Categories and quantities of object images presented to subjects in the behavioral IP test. The IP test consists of 13 unique objects images (visual stimuli) in each of the four categories – man-made power, natural power, man-made precision, natural precision – describing the characteristics of each object. Additionally, every object is presented in two orientations – horizontal and vertical. Every image also contains an arrow indicating to the subject the place on the object to imagine forming the grip around. There are a total of 104 visual stimuli presented to the subject in a single block. Images were presented in a random order.

Once the IP test is started by the researcher, a brief loading screen appears. Then there are some short instructions, during which the subject must calibrate the glove. (Full instructions about the test were provided to the subject beforehand.) The calibration is done by extending and flexing all fingers a few times as well as adducting and abducting all the fingers a few times. This step is necessary to set the maximum and minimum output values for the sensors. Since hand sizes and range of motion can vary between subjects, it is important to calibrate the gloves as a means of somewhat normalizing the results. The subject can begin the test once they are ready by pressing any button. A single block consists of 104 trials. The format of each trial is shown in Figure 3-5. Each

trial consists of the interstimulus interval (ISI), fixation point, object stimulus presentation, and grip formation cue, and lasts 4 to 4.5 seconds. The ISI is randomized between 500 and 1000 milliseconds to prevent the subject from falling into a predictable pattern. During this phase, the subject sees a blank gray screen. The 500 milliseconds fixation screen presents a red plus in the middle of the screen to focus the subject's vision. Then object image is displayed on the screen for 1500 milliseconds. During this time, the subject hand is still in the resting position as they only plan the grip they will make. After the stimulus, a screen with the text "Form grip" appears for 1500 milliseconds. This is the cue for the subject to make the grip. They are instructed to raise their hand from the resting state, form the grip, and return to the resting state – all within the 1500 milliseconds as much as they possibly can. Limiting the handgrip motion to a specified time window allows for simpler data analysis as well. There is a user-controlled rest phase halfway through each block, to reduce fatigue during the test. The subject can rest for up to one minute or continue at any time with a button press. With 104 trials, the test lasts approximately 7 to 8 minutes for a single block.

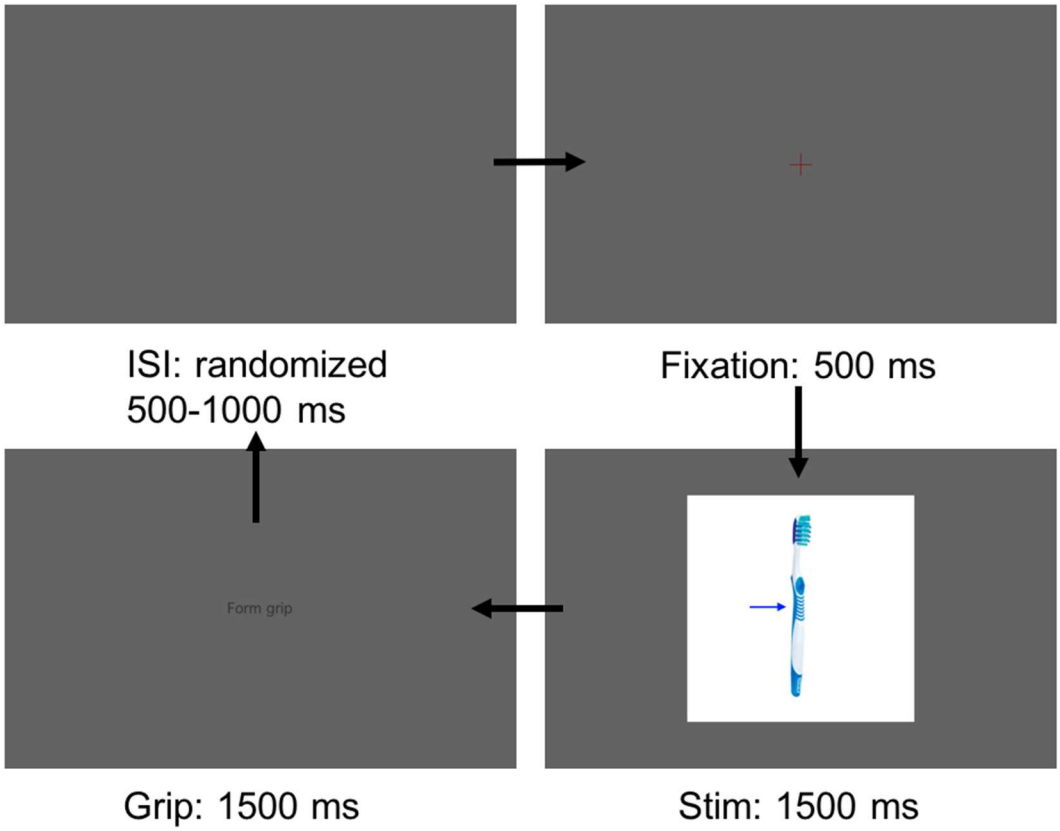

Figure 3-5: Time course for a single trial in the behavioral IP test. Each trial begins with an interstimulus interval randomized between 500 and 1000 ms, during which the subject sees a blank gray screen. This is followed by the fixation point for 500 ms, during which the subject was instructed to focus on a red cross at the center of the screen. Next, the visual stimulus is presented for 1500 ms, during which the subject observes the object image and decides the grip to form. The image presented is selected randomly from the 104 visual stimuli. Lastly, the "Form grip" text appears on the screen for 1500 ms, during which subjects were instructed to lift their right hand from the resting position, form the grip that they would to grip the object, and return to the resting position. After this, the next trial begins.

Before administering the IP test for behavioral data, a framework was put in place to test the proper function of the handgrip tracking components within the system. Healthy participants wore the gloves with accelerometers and took the preliminary version of the test to validate that proper handgrips are recognized and that the paradigm is appropriate. Additionally, the objects were validated before running any tests. Four

healthy volunteers (3 males, 1 female, age range 21–29 years, mean = 27.56, SD = 4.03) with no history of neurological or psychiatric disorder were presented with the object images and asked to categorize them into power or precision based on the grip they would use to grasp the object. A button press (1 or 2) recorded their response. No glove or accelerometer data were collected since no grip was formed during this validation. The objects that were incorrectly categorized during validation were removed. The purpose of the validation was to ensure that subjects would form the expected power or precision grip. Of course, this may vary for a few object images due to personal interpretation or familiarity, but we attempted to reduce ambiguity when selecting the objects.

3.2.5 Data acquisition

Since the behavioral test must run simultaneously alongside the glove and accelerometer data collection, the parallel computing tools available in Matlab were used. Specifically, the "batch" command was called on to first run the scripts that record data from the glove and accelerometer. Then BraviShell was initialized to run the behavioral test while glove and accelerometer data collection was working in the background. A pre-defined amount of time has to be set in the glove and accelerometer data collection scripts. This is determined by the user based on the duration of the experimental paradigm. Output data from both the glove and accelerometer are saved in the Matlab workspace and as .xls files. The parameters and results (timing, object categorization) of the IP test are also exported to an .xls file.

Although a pair of accelerometers and gloves were set up to interface, only the right hand was used for this test. The subject wore the glove and the accelerometer was

taped to the dorsal side of the glove, just below the knuckles. This position would account for whole hand movements rather than one part. Prior to performing the test, subjects were verbally given full instructions and practiced the test on some practice trials until they felt comfortable with the paradigm. The test was administered on Matlab R2015b (MathWorks, Natick, MA, USA) on an Apple MacBook Pro Retina 15-inch computer.

3.2.6 Data analysis

The times and data recorded by the accelerometers and gloves were saved in an .xls file. However, after each subject, the data was stored in the Matlab workspace and manually saved as .mat files as well. The data was saved as a time series for the entire session and then resampled and organized into per trial time series for each subject. For the accelerometers, this was a 3-by-136-by-*n* matrix, where *n* is 208 trials for subjects that had two blocks and 312 trials for subjects with three blocks. There were 3 axes and 136 samples per trial, since the data was resampled to 45 Hz and the time epoch selected was three seconds – 1.5 seconds of the visual stimulus and 1.5 seconds of the grip formation. For the gloves, the matrix dimensions were 14-by-196-by-*n* since there are 14 sensors and the sampling rate was set to 65 Hz. The reference point for the three-second time epoch was the time when the visual stimulus was presented, which was saved in the behavioral test output file.

The 3D matrices from each of the two data sources were also categorized. For the accelerometers, the *n* trials were split into power and precision grip objects (*n*/2 each) and also horizontal and vertical grip orientations (*n*/2 each). For the gloves, the data was only

sorted by power and precision (n/2 each), since the glove data could not provide meaningful information about hand orientation. The categories assigned to each trial were extracted from the behavioral test output file. It should be noted that this categorization was what each object was assigned prior to the test and not the response from the subject. As mentioned earlier, the objects selected were validated to attempt to reduce ambiguity in whether it should be a power or precision grip. However, we do attempt to use a method (described later) to classify the recorded glove and accelerometer data into these categories.

Although the raw accelerometer data did not provide much insight on its own, we were able to use the z-axis of the accelerometer to look at the movement of the subject's hand as they lift it from the resting state to begin forming the grip and subsequently return to the resting state. The more interesting information, however, was seen in the roll and pitch angles calculated from the three axes of the accelerometer data. These measurements are based on the constant acceleration due to gravity acting on the accelerometer. Using trigonometry to calculate the vector components of gravitational acceleration, the angle of tilt can be determined in two degrees of freedom (Pedley, 2007). The equations for roll and pitch from the x, y, and z accelerations are as follows:

$$Roll\colon \phi = \tan^{-1}\frac{G_y}{G_z} \qquad Pitch\colon \theta = \tan^{-1}\frac{-G_x}{\sqrt{{G_y}^2+{G_z}^2}}$$

G_x, G_y, and G_z represent the accelerometer outputs for their respective axes minus an offset value to normalize them around 0. Subtracting the offset redefines G_x and G_y to approximately zero, while G_z represents the acceleration from gravity. The offset was

calculated from the accelerometer reading at rest. Once calculated, roll angle is a direct measurement of wrist rotation, which indicates whether the hand was in a pronated or neutral position – the two orientations we're studying. The pitch angle tells us how far forward the hand was tilted. Roll and pitch angles were calculated for the entire continuous time series as well as the per trial time series for each subject. In addition, roll angle was used to calculate the angular displacement from resting position to maximum rotation for each trial. The angular displacement, which is a measure of change from initial position to final position, allows us to compare between subjects, since the resting position angle was not uniform for all subjects.

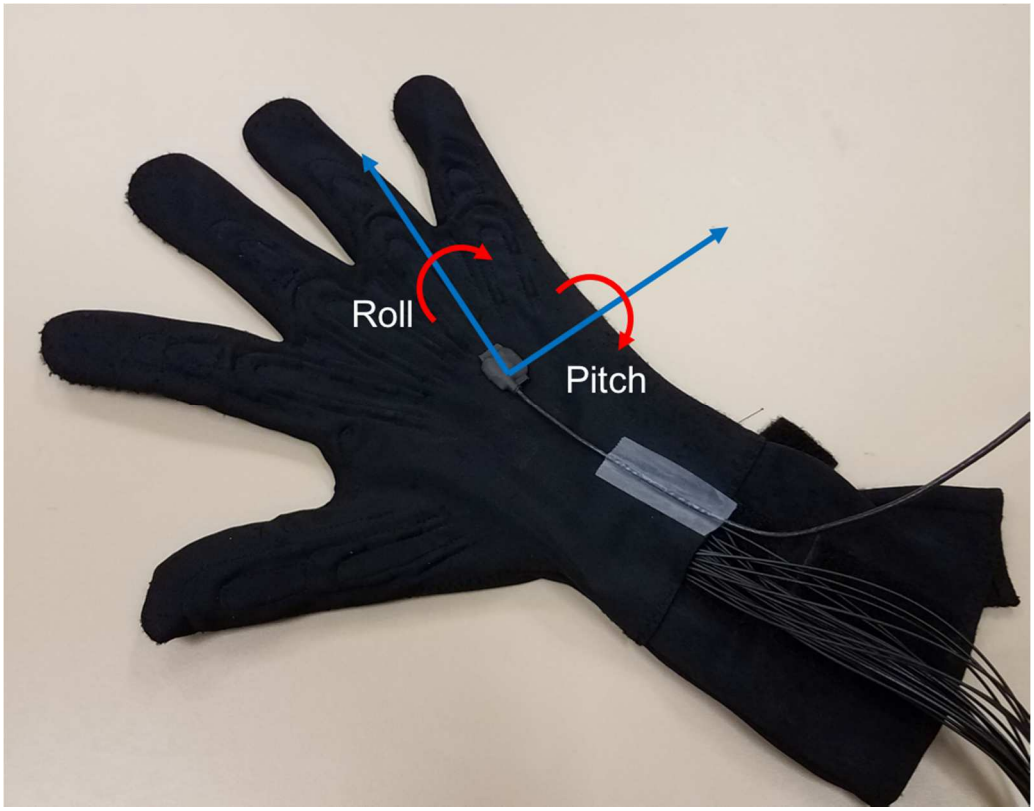

Figure 3-6: Axes along which the roll and pitch angles are measured on the glove and accelerometer. Although not shown here, the accelerometer was taped down to the glove on the back of the subject's hand to prevent it from changing position. Roll angle corresponds to the rotating the wrist while pitch angle corresponds to tilting the hand forward or back.

The raw glove data also had to be processed before applying any statistical methods to compare subjects. Although the 14 sensors do provide useful information about the movement of each finger, we wanted to reduce the number of dimensions to be able to characterize the entire handgrip in one dimension. To achieve this, principal components analysis (PCA) was applied to the glove data. PCA is a method that takes a dataset of observations with several correlated dependent variables and transforms them onto a new set of orthogonal variables (Abdi and Williams, 2010). In this case, the 14 sensors were reduced to a single set of observations over time. The PCA result made for simpler analysis within subjects and comparisons between subjects.

As described earlier, the accelerometer and glove data were categorized into power and precision as well as horizontal and vertical based on which category each object was assigned. However, while we attempted to reduce ambiguity, we still expect there to be some deviation between assigned category and actual subject response, due to individual biases or interpretations of the object. To classify the data based on the measured signals, *k*-means clustering was implemented to categorize the accelerometer data into horizontal and vertical and the glove data into power and precision. *k*-means clustering takes a set of observations and partitions them into clusters based on each observation's distance to the centroid (nearest mean of the points in that cluster). In this case, the input data was each trial's three-second time epoch. For the accelerometer, we wanted to form two clusters based on the wrist rotation angle, so the input was the time series of the roll angle calculation, not the raw accelerometer data. For the glove, the two desired clusters were formed based on the time series of the first principal component.

The distance between each observation and its centroid was the squared Euclidean distance. Unlike our previous categorization, clustering did not return an equal number of observations in each category. To have a better understanding of whether this clustering strategy was appropriate, the percentage accuracy of the k-means clustering was determined by comparing its categorization results to the assigned category of each object. It should be noted that this accuracy would also be affected by whether the subject formed the intended grip for a given object.

Another metric used to characterize hand movements was the onset timing of each trial. Onset timing is a measure of the start of the behavioral response, which, in this case, is a particular hand movement relative to the presentation of the "form grip" cue. Onset times were selected as the first time point at which the amplitude of the signal exceeds 3.1 standard deviations (Letham and Raij, 2011) from the resting phase mean value, indicating when the motion begins. Two sets of onset timing values were calculated. For the accelerometer data, the raw z-axis acceleration was used to determine the onset of the subject's hand being lifted up from the resting state. These onset times were then categorized into power and precision. With the glove data, each trial's first principal component signal was used to determine the onset time for the grip being formed, and also categorized into power and precision.

3.3 Results

3.3.1 Object categories

The percentage of objects categorized correctly by each participant in the validation test is shown in Table 3-3. Between 30 and 32 distinct power grip objects were

presented to each participant and between 29 and 33 distinct precision grip objects were presented to each participant. Objects that were categorized incorrectly by multiple subjects were ultimately removed from the test. Based on these validation results, we finally included 52 total objects, or 26 each of power and precision grip objects.

	FY	JY	KR	YG	**TOTAL**
Power	100.00%	87.10%	90.00%	93.75%	**92.74%**
Precision	90.32%	96.55%	86.21%	72.73%	**86.07%**

Table 3-3: Percent of object images correctly categorized during validation.

3.3.2 Raw accelerometer data

An example of the raw accelerometer data in three axes is shown in Figure 3-7. This was collected from a single subject's session with three blocks (312 trials). The x-axis is the time course for the three blocks, in seconds, and the y-axis is the voltage output representing acceleration. Each vertical red line represents the time point at which a visual stimulus was presented to the subject. These event markers are used to categorize each trial into time windows.

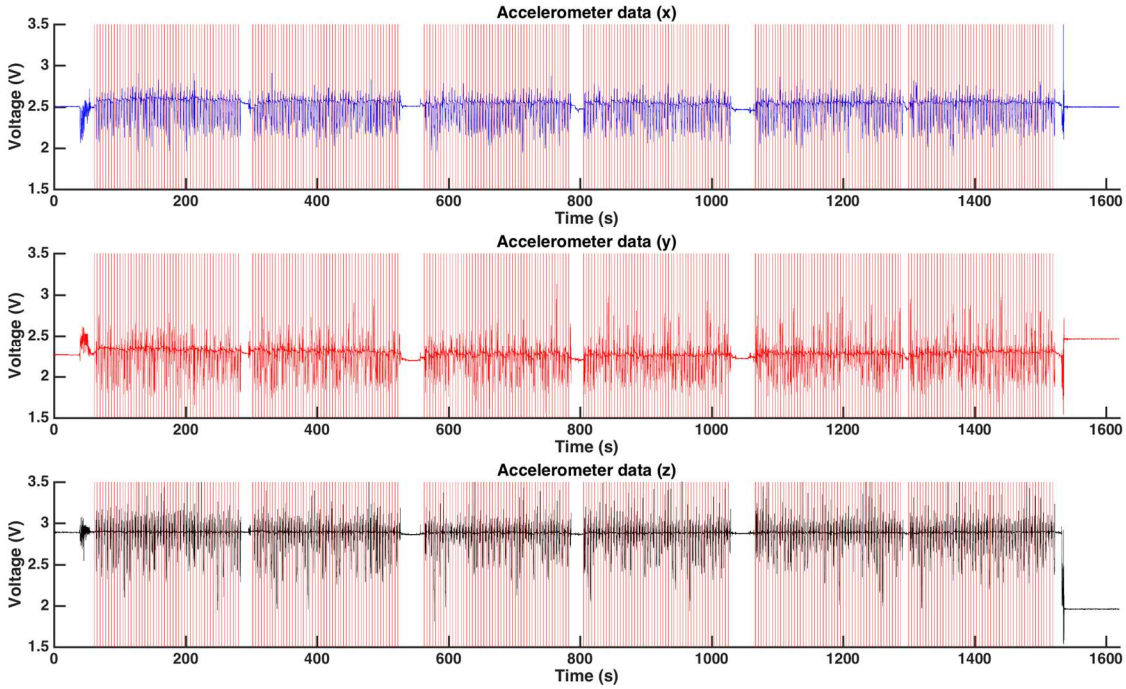

Figure 3-7: Raw accelerometer time series data from three channels (x, y, and z axes) during three IP test blocks in one subject. The x-axis in each plot represents the time in seconds as three blocks were run consecutively. The y-axis in each plot represents the voltage output reading from the respective axis of the accelerometer. The vertical red lines in each plot are the stimulus markers, indicating the time at which the visual stimulus was presented to the subject on the screen.

While the x and y axes readings from the raw accelerometer data do not provide us with any substantial information, we can use the z-axis time series to determine the onset timing for hand movement. Figure 3-8 is an averaged single trial z-axis accelerometer signal for a single subject. The time on the x-axis from -1.5 seconds to 0 seconds is the subject's hand is in a resting (pronated) position during the visual stimulus presentation on the screen. At 0 seconds, the grip cue appears on the screen and the subject begins the motion of forming the grip. The y-axis indicates the output voltage reading, corresponding to acceleration. The average signal is calculated for the power grip trials (show in red) and precision grip trials (shown in blue). The z axis acceleration

is of particular interest since the subject raises their hand for each trial, and the change in amplitude is evident from the averaged signal. The amplitude is negative since the positive z-axis is facing downward, in the direction of gravitational acceleration.

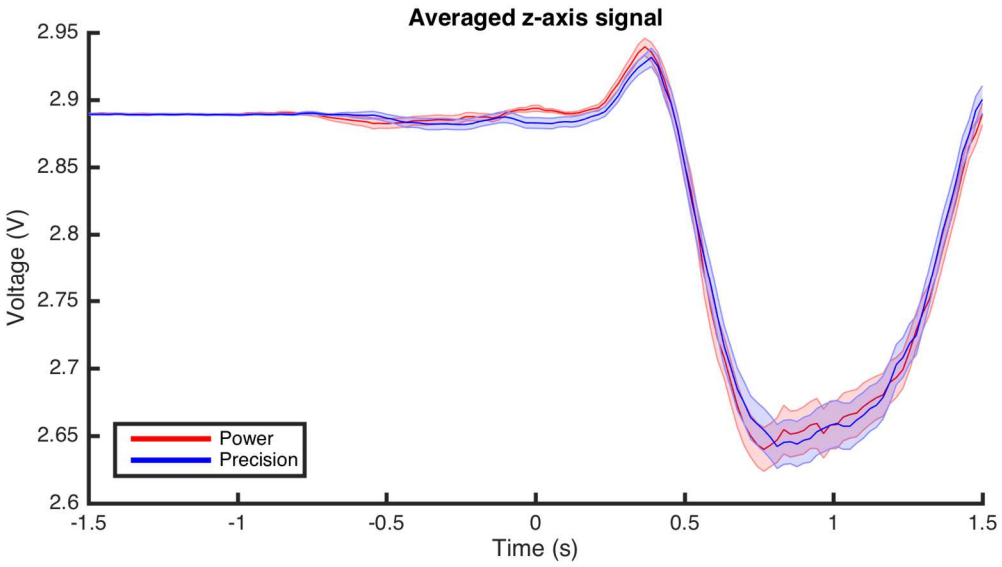

Figure 3-8: Average z-axis accelerometer data during one trial in one subject, categorized into power and precision grip trials. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the voltage reading. This plot illustrates the reading from only the z-axis channel on the accelerometer. The red line represents the mean for the trials categorized as power grip and the blue line represents the mean for the trials categorized as precision grip. Both lines contain shaded areas representing their respective standard error.

Onset timing is calculated from z-axis acceleration for each trial for each subject and categorized into power and precision grip. Figure 3-9 shows the distribution of onset times for all trials across all subjects. There were 1797 trials for power grip and 1813 for precision grip, out of 1924 total trials for each. Trials for which an onset timing value could not be calculated were excluded. (In these trials, the amplitude of the signal never exceeded the designated 3.1 SD threshold.) The x-axis represents the amount of time

after the grip formation phase has begun – it is the same timescale as Figure 3-8. The mean onset time for the power grip trials was 0.42 seconds (SD = 0.22) and the mean onset time for the precision grip trials was 0.40 seconds (SD = 0.23). A Kolmogorov-Smirnov comparison test was run to check for a significant statistical difference between the two groups. There is a statistical difference between the two distributions (p-value of 0.0249).

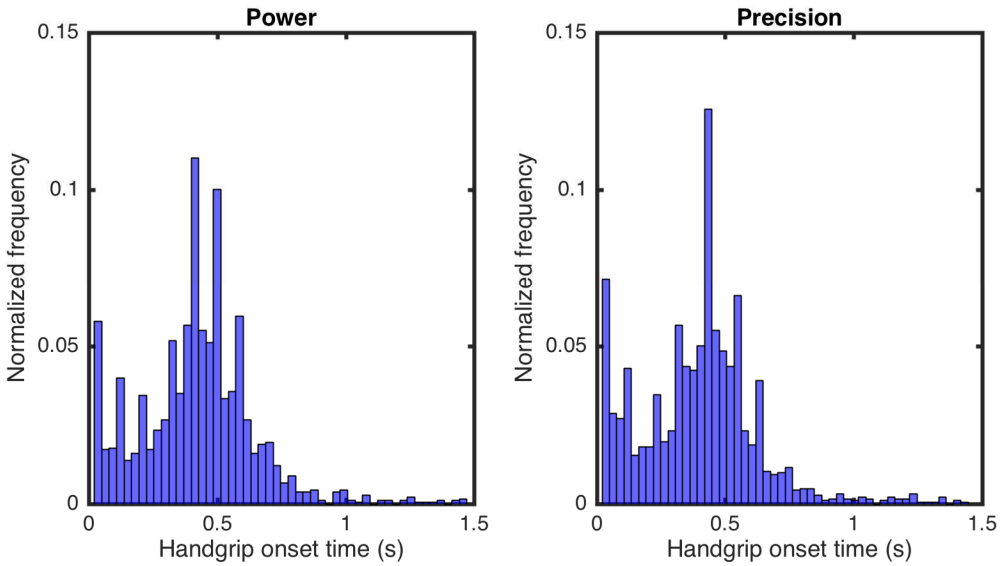

Figure 3-9: Distribution of onset times across all trials in all 15 subjects, as calculated from z-axis acceleration, for power and precision grip trials. The x-axis in each histogram shows the amount of time in seconds after the "Form grip" cue is presented to the subject. The y-axis in each histogram is the proportion of samples from the total sample size for the respective category. The power grip trials (histogram on left) has a sample size of 1797 and the precision grip trials (histogram on right) has a sample size of 1813. The onset time represents the amount time, after being cued, that it takes the subject to start making the handgrip response. Both distributions are centered around approximately 0.45 seconds, with a peak near 0 seconds as well.

3.3.3 Angle of rotation

We were able to calculate roll and pitch angles using the raw accelerometer data. An example of these calculated angles from a single subject is shown in Figure 3-10. For roll, a positive angle represents clockwise wrist rotation, and for pitch, a positive angle

represents tilting the hand forward. Once again, each red vertical line indicates the time points at which a visual stimulus appeared on the screen and was used for separating the entire time series data into trials.

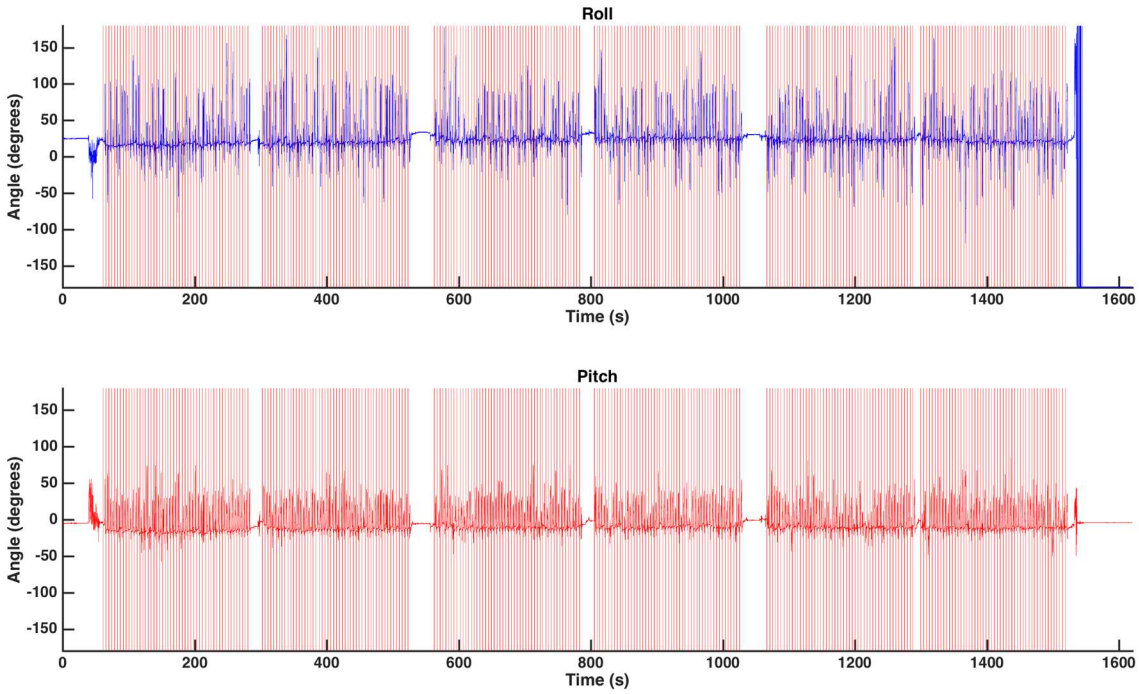

Figure 3-10: Roll and pitch angles calculated from accelerometer data during three blocks in one subject. The x-axis in each plot shows the time in seconds as three blocks were run consecutively. The y-axis in each plot shows the angle of rotation in degrees. For the roll measurement, a clockwise wrist rotation represents an increase in angle. For the pitch measurement, tilting the hand forward represents an increase in angle. The vertical red lines in each plot are the stimulus markers, indicating the time at which the visual stimulus was presented to the subject on the screen.

Each trial for all subjects was averaged by hand orientation category and the single trial average for roll angle shown in Figure 3-11. The number of trials was the same for each category. Roll is of more interest than pitch since it is a direct measurement of wrist rotation, which represents the two orientation positions being studied. The time axis remains the same as in Figure 3-8. The y-axis is the normalized angular wrist rotation, with the resting position set to 0 degrees for each trial. Although

the resting position is pronation, there is still a slight rotation for objects designated as horizontal orientation, or pronation position. For objects that are vertical orientation grip, or the neutral position, there is a clear wrist rotation across subjects.

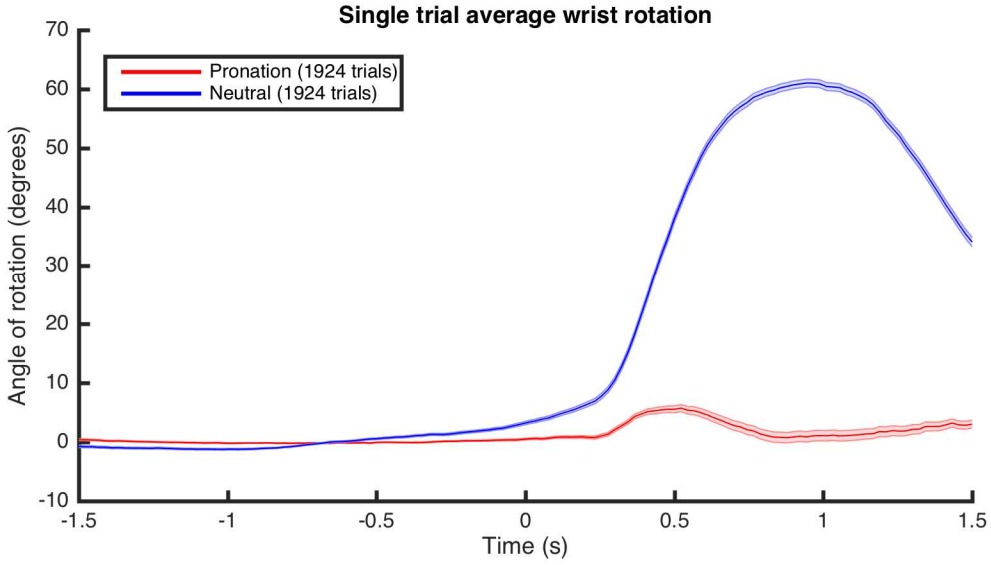

Figure 3-11: Average roll angle during one trial across all subjects, categorized into pronation and neutral position trials. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the roll angle measurement (which we are using as a correlate for wrist rotation) in degrees. A clockwise rotation is represented by an increase in angle. The red line is the mean of all the trials categorized as pronation position (n=1924) and the blue line is the mean of all the trials categorized as neutral position (n=1924). Both lines contain shaded areas representing their respective standard error.

From this averaged wrist rotation measurement, we wanted to compare the results across all subjects for each category. The maximum angular displacement in wrist rotation was calculated for each trial across all subjects. This value may be positive (for a clockwise wrist rotation) or negative (for a counterclockwise rotation). The highest angular displacement value was taken for each trial and the distribution is shown in Figure 3-12, categorized into horizontal (pronation) and vertical (neutral) object groups.

The mean pronation position angle is 0.121 degrees (SD = 53.93) and the mean neutral position angle is 71.68 degrees (SD = 34.85).

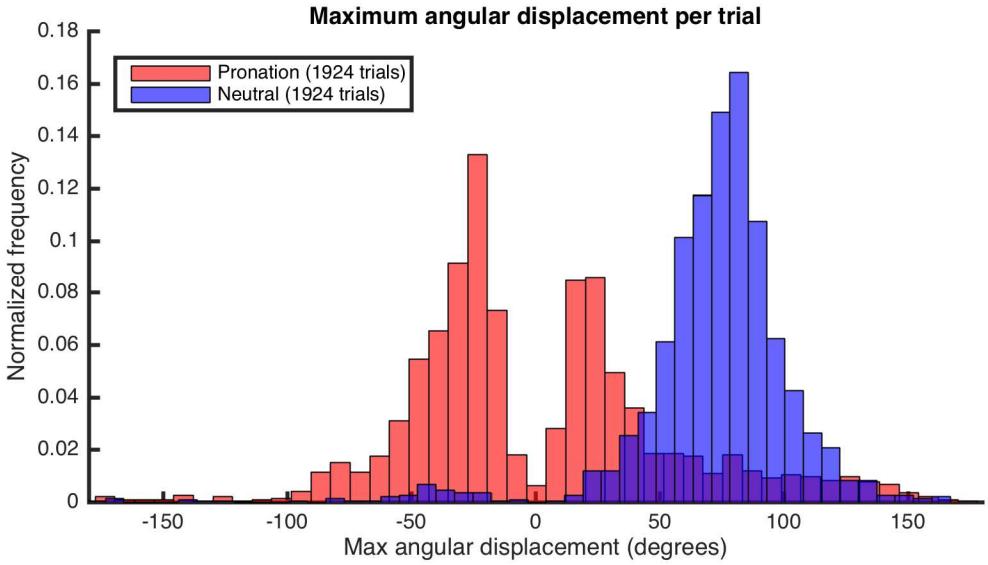

Figure 3-12: Distribution of maximum angular displacement across all trials in all 15 subjects, calculated from roll angle, categorized into pronation and neutral position trials. The x-axis shows the angular displacement in degrees. Maximum angular displacement is calculated as the difference between mean roll angle at resting position and maximum absolute value of roll angle during grip formation, for each trial. This represents the greatest magnitude of wrist rotation (clockwise or counterclockwise) during each trial. The y-axis shows the proportion of samples from the total sample size for the respective category. The distribution of pronation position trials (in red, n=1924) has a peak at approximately -25 degrees and another at approximately 25 degrees. The distribution of neutral position trials (in blue, n=1924) is centered around approximately 75 degrees.

Up until this point, the objects have been grouped according to categories assigned before the test is administered. However, this does not account for the subject's measured hand movement. For wrist rotation, the objects are presented at a certain angle, and the subject is expected to rotate their hand to make the grip at that angle. In order to categorize the measured data, we used the wrist rotation angles calculated from the accelerometer data to classify each trial into horizontal or vertical. A *k*-means clustering algorithm was implemented to form two groups. Out of the 1924 trials originally

categorized as pronation position objects, 85.9% were correctly identified in the clustering. This value was 90.5% for neutral position objects, which also consisted of 1924 trials. This accuracy measurement is the percent of trials that had the same categorization in the original object categories as well as the clustered categories, but does not account for wrist rotations that did not match the intended orientation.

In order to have a clearer understanding of the clustering results, the single trial average that was previously reported was plotted again in Figure 3-13, but categorized by the clustering results. The data was clustered as 1836 trials of pronation position and 2012 trials of neutral position. The axes remain the same as in Figure 3-11.

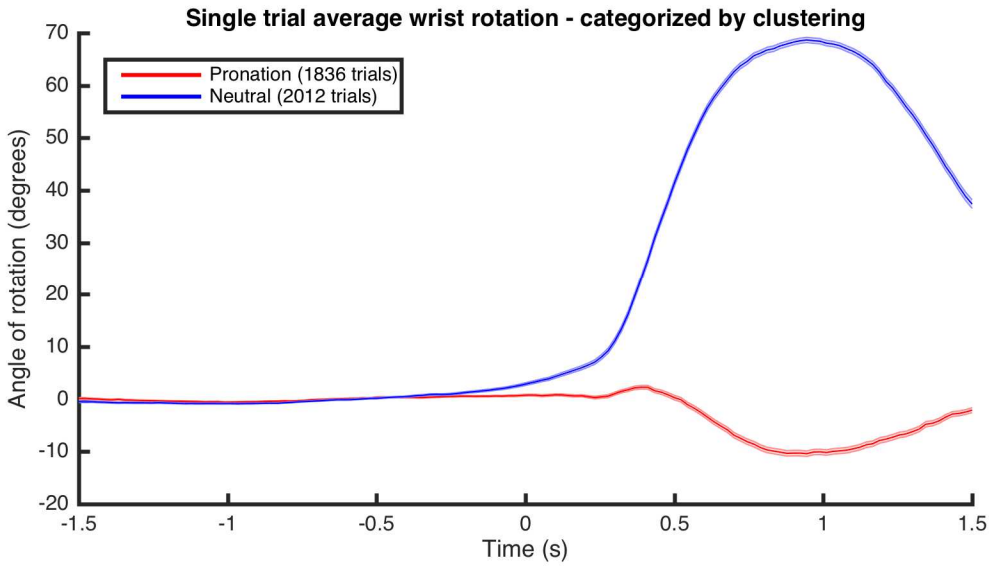

Figure 3-13: Average roll angle during one trial across all subjects, categorized into pronation and neutral position trials based on clustering results. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the roll angle measurement (which we are using as a correlate for wrist rotation) in degrees. A clockwise rotation is represented by an increase in angle. The red line is the mean of all the trials categorized as pronation position (n=1836) and the blue line is the mean of all the trials categorized as neutral position (n=2012). Both lines contain shaded areas representing their respective standard error.

3.3.4 Glove data

The glove data consists of the 14 sensors with each value between 0 and 1. An example of the glove data for a single subject over one block (104 trials) is shown in Figure 3-14. The x-axis is the time for this block within the course of the entire session for the subject. Along the y-axis, the signals from all 14 sensors are represented separately. Lower values represent finger extension and higher values finger flexion. The sensors are along the back of the hand, and 1 to 14 are located from the thumb to the pinky. A detailed description of the sensor placements was provided in a prior chapter.

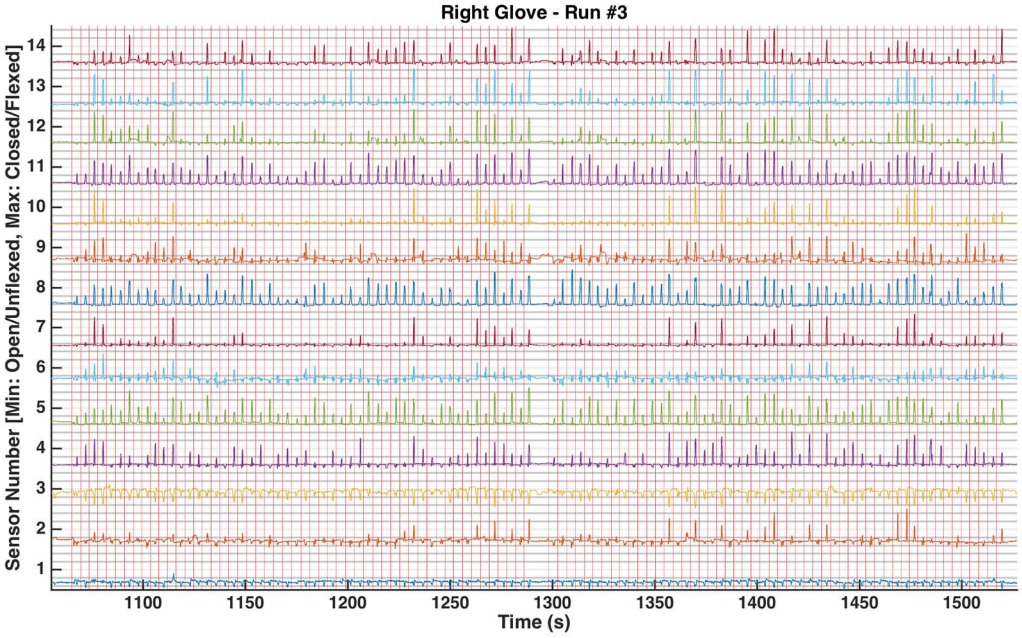

Figure 3-14: Data glove recordings from all 14 sensors during one IP test block in one subject. The x-axis represents the time in seconds during the course of one block. Time does not start at 0 seconds since this is one block within a series of blocks. The y-axis contains 14 separate time series plots, each representing an individual glove sensor. The sensors are located on the along the back of the glove, with two flexion sensors on two joints at each finger and one abduction sensor between each finger. Sensors are numbered 1 to 14 in order from left to right, starting at the base of the thumb and ending at the second joint of the pinky. For each sensor, the output value is between 0 and 1, with higher values representing flexion or adduction, depending on sensor location. In certain sensors, differences are visible based on grip type. The vertical red lines are the stimulus markers, indicating the time at which the visual stimulus was presented to the subject on the screen.

Although the glove data can describe the individual finger movements, we are interested in characterizing whole handgrips. The 14 sensors were reduced to a single dimension, while still accounting for the contribution of all sensors, by conducting PCA. Figure 3-15 shows the same glove recordings from Figure 3-14 after PCA has been applied. A separate PCA was calculated for each subject, providing us with a time series in a single dimension describing the glove data. As with the accelerometer data, the glove PCA data was also organized by trial and categorized. Although the same was done for the raw glove data, it would not be appropriate to compare or calculate the mean for each sensor's raw data across the 14 subjects since there is significant variability in recordings between subjects.

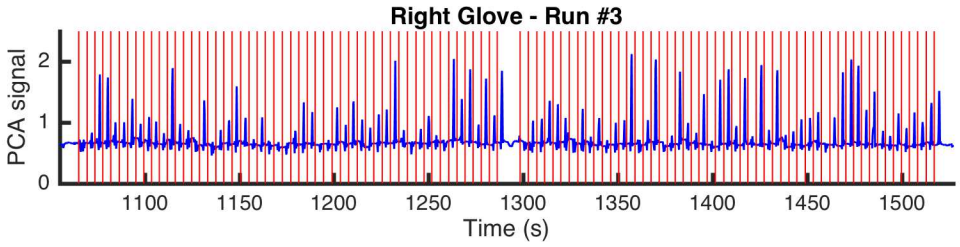

Figure 3-15: Principal component 1 during one block in one subject, calculated from 14 sensor glove data. The x-axis represents the time in seconds during the course of one block. Time does not start at 0 seconds since this is one block within a series of blocks. The y-axis shows the magnitude of flexion or adduction, with increasing values representing further finger flexion or adduction. Since this is described by a principal component, the y-axis value is a combination of the flexion and adduction sensor outputs, and does not completely represent one or the other. The vertical red lines are the stimulus markers, indicating the time at which the visual stimulus was presented to the subject on the screen.

The first principal component signal for each trial was categorized into power and precision grip objects and the mean time series calculated across all subjects. The results are shown in Figure 3-16. The x-axis timing remains the same as the accelerometer data, with -1.5 to 0 seconds representing the visual stimulus phase and 0 to 1.5 seconds

representing the grip formation phase in each trial. The total number of trials in each category was 2496. The amplitude of the first principal component for an object categorized as power grip is greater than that of an object categorized as precision grip.

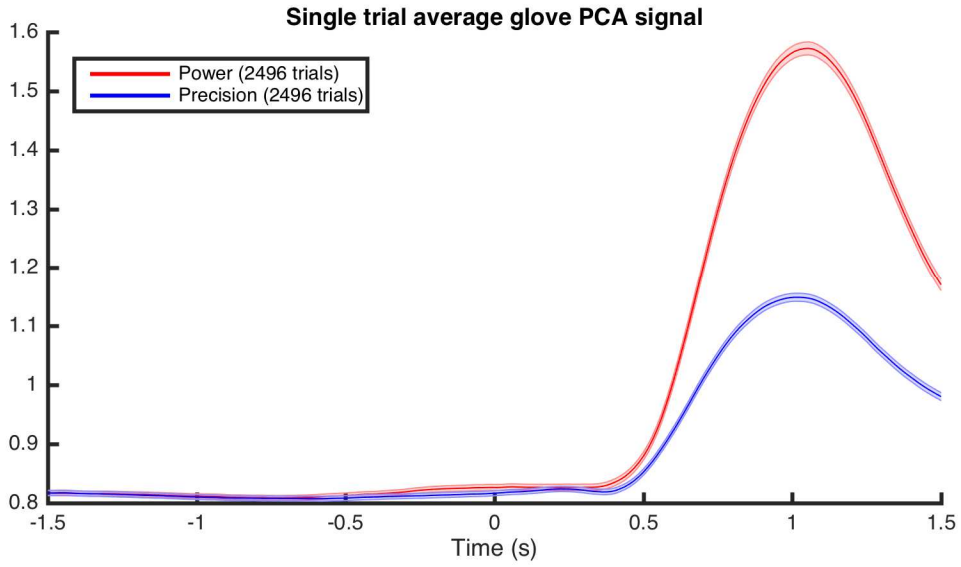

Figure 3-16: Average principal component 1 signal during one trial across all subjects, calculated from 14 sensor glove data, categorized into power and precision grip trials. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the magnitude of flexion or adduction, with increasing values representing further finger flexion or adduction. Since this is described by a principal component, the y-axis value is a combination of the flexion and adduction sensor outputs, and does not completely represent one or the other. The red line is the mean of all the trials categorized as power grip (n=2496) and the blue line is the mean of all the trials categorized as precision grip (n=2496). Both lines contain shaded areas representing their respective standard error.

With the accelerometer data, we had calculated onset timing from the z-axis acceleration, representing the time at which the subject began the motion of lifting their hand. With the glove data, we can calculate onset timing for the grip formation. The first principal component time series for each trial across all subjects was used to determine onset timings. To do so, we first calculated the PCA separately for each subject, rather

than pooling all the trials together first, as was done with previous analyses. This was done since the PCA for each subject would differ with respect to the relative contribution of each sensor. Then, the onset times were calculated for each trial and were all compared. The distribution of onset timings for grip formation by power and precision is shown in Figure 3-17. On the x-axis, 0 seconds represents the start of the grip formation cue appearing on the screen. The power category consists of 2323 trials and the precision consists of 2138 trials. The total 2496 trials are not reported for each category since the signal did not reach the minimum threshold for establishing onset time in all cases. The mean onset time for power was 0.55 seconds (SD = 0.26) and the mean for precision was 0.53 seconds (SD = 0.29). A Kolmogorov-Smirnov comparison test was run to check for a significant statistical difference between the two groups. There is a statistical difference between the two distributions (p-value of 0.0038).

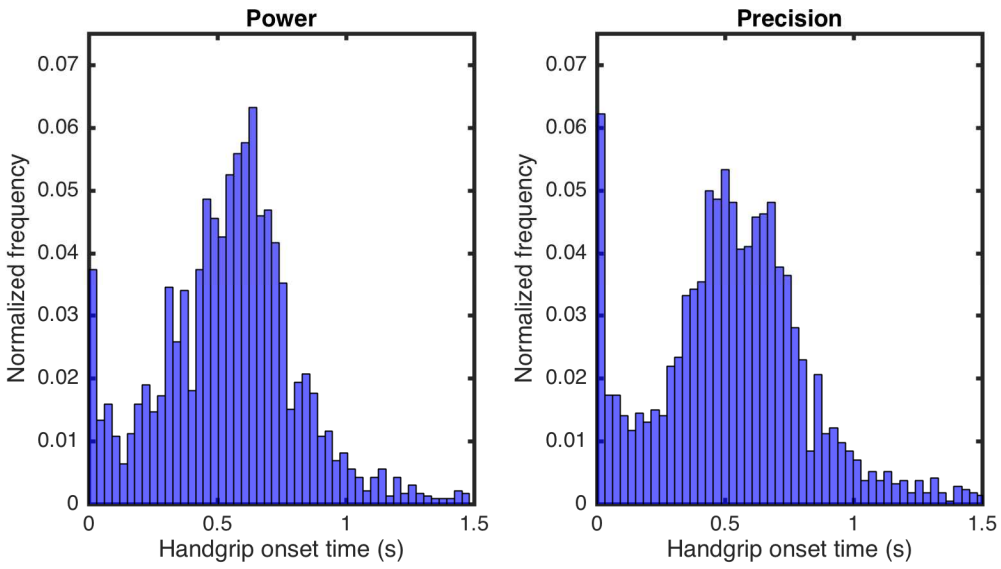

Figure 3-17: Distribution of onset times across all trials in all 20 subjects, as calculated from data glove principal component 1 signal, for power and precision grip trials. The x-axis in each histogram shows the amount of time in seconds after the "Form grip" cue is presented to the subject. The y-axis in each histogram is the proportion of samples from the total

sample size for the respective category. The power grip trials (histogram on left) has a sample size of 2323 and the precision grip trials (histogram on right) has a sample size of 2138. The onset time represents the amount time, after being cued, that it takes the subject to start making the handgrip response. Both distributions are centered around approximately 0.55 seconds, with a peak near 0 seconds as well.

The power and precision categories used thus far have been based on the categories assigned to each object. As with the accelerometer data, we applied a *k*-means clustering algorithm to distinguish between power and precision grip from the glove readings. Clustering was done on the first principal component signals. Clusters were formed for each subject and then the trials from each cluster across all subjects were grouped together. From the 2496 trials originally categorized as power, 68.1% were recognized as power from the clustering. This value was 83.4% for precision, also 2496 trials. The trials were averaged and visualized, seen in Figure 3-18. This plot contains all the same trials as Figure 3-16, but is categorized based on the clustering results. 2114 trials were categorized as power grip and 2878 trials categorized as precision.

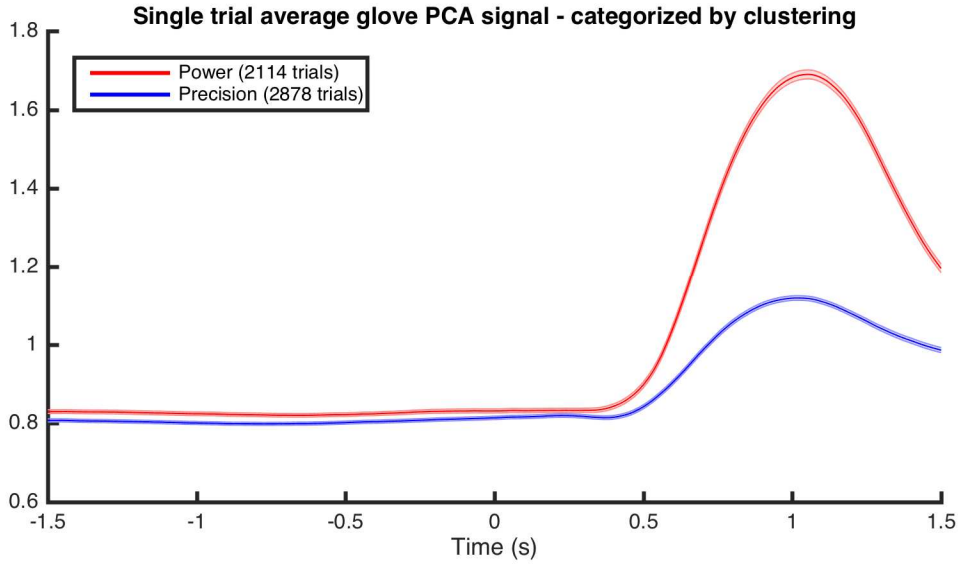

Figure 3-18: Average principal component 1 signal during one trial across all subjects, calculated from 14 sensor glove data, categorized into power and precision grip trials based on clustering results. The x-axis shows the time during a single trial, with 0 seconds

representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the magnitude of flexion or adduction, with increasing values representing further finger flexion or adduction. Since this is described by a principal component, the y-axis value is a combination of the flexion and adduction sensor outputs, and does not completely represent one or the other. The red line is the mean of all the trials categorized as power grip (n=2114) and the blue line is the mean of all the trials categorized as precision grip (n=2878). Both lines contain shaded areas representing their respective standard error.

3.4 Discussion

The objective of the first part of this project was to demonstrate proper functionality of the individual components of the system. In particular, we wanted to show that meaningful information could be extracted from the accelerometer and glove data. Many of the results obtained, from both the raw data and the calculated measurements, successfully show that the recordings from these sensors are as expected.

From the raw accelerometer data, the z-axis was used to calculate onset timing. The results of onset timing across all trials in Figure 3-9 indicate that although the mean and standard deviation are similar for both groups, there is a statistical difference between the two groups. Using the Kolmogorov-Smirnov test, we were able to reject the null hypothesis that the two datasets are from the same distribution. This evidence suggests that in this set of samples, subjects may be starting their handgrips for power and precision at different times. However, this assumption relies on all the conditions used to calculate onset timing. For example, the threshold selected for onset timing was 3.1 standard deviations beyond the mean of the resting phase signal. Adjusting this threshold or incorporating additional conditions may result in different outcomes for the comparison test.

The measurement we were interested in from the raw accelerometer data were the roll and pitch angles, calculated from the raw accelerometer values. Specifically, the roll angle is a direct measurement of the wrist angular rotation. The average wrist rotation angle over the course of a single trial was illustrated in Figure 3-11. We see that there is a clear difference between the two categories. For the objects categorized as horizontal (pronation) we would expect the angle of rotation to remain steady from the resting phase to the grip formation. The 1924 pronation trials averaged in the plot indicate as such, with the exception of a very slight (<5 degrees) initial clockwise rotation followed by a return to the original orientation. This can likely be attributed to a natural motion as the hand is lifted up to form the grip. For the vertical (neutral) objects, subjects are rotating their hand about 90 degrees clockwise before forming the grip, followed by a return to the resting position before the start of the next trial. In the plot, we see that the average time series from the 1924 neutral trials represent the expected behavior. The angle of rotation steadily rises to approximately 60 degrees over 0.5 second and then steadily returns to 0 degrees.

Figure 3-12 is a distribution of the maximum angular rotations for all trials. From the results of the averaged single trial and our expectations about the subject's wrist rotation, we would expect the maximum angles for pronation trials to be focused around 5 degrees and those of the neutral trials around 60 to 70 degrees. The histogram shows two peaks for the pronation objects, one below 0 degrees and another above 0 degrees. There is a dip at 0 degrees since 0 degrees represents the mean angle during the resting phase. Any hand movement results in even a slight rotation away from resting position in

either direction. The mean angle is 0.121 degrees, which is reasonable. However, the standard deviation of 53.9 degrees is fairly high. The distribution contains several points that are much higher and much lower than 0 degrees. Outliers can be attributed to trials in which the subject rotated their wrist differently from what was intended for that particular object image. Additionally, the value being looked at is the maximum angle for each trial, which may not necessarily be a proper characterization of the wrist rotation across the entire trial in some instances. For the neutral position group, the mean angle is 71.68 degrees, which is a reasonable value as well. The standard deviation, at 34.85 degrees, is less than that of the pronation group. The values at either extreme of the distribution can be accounted for by the same reasons as the pronation group.

The basis for comparisons between groups has been the category we assigned to each object. Whatever the object category for a given trial, we assumed would also be the type of handgrip and orientation made by the subject, with perhaps a few exceptions. Our results did support this for the most part, as the difference in wrist rotation was evident between the two groups. However, we carried out *k*-means clustering as a means of categorizing trials based on the actual measured results. The percent accuracy value represents the percent of trials from the total that were assigned the same category after clustering, for each category. Using the wrist rotation angle time series to perform the clustering, 85.9% were correctly identified pronation and 90.5% correctly identified neutral. For the trials that were incorrectly categorized the subject may have been performing the opposite wrist rotation, in which case the clustering algorithm is correct, but the subject's response was not as expected. The time series of the averaged trial data

by the clustering categories (Figure 3-13) is consistent with the expected results.

The average principal component 1 signal from the glove data is shown in Figure 3-16. The difference between power grip and precision grip is visually evident, with the power grip PCA signal having a greater amplitude than that of the precision grip. This averaged principal component 1 signal includes a contribution from all 14 sensors. A higher amplitude in one group compared to the other indicates that certain sensors consistently had higher amplitudes for power grip. This is the expected result, since flexion of the fingers increases the sensor reading. Thus, if more fingers are flexed or the amount of flexion is greater for select fingers during power grip, the expected signal would have a higher amplitude for power grip trials.

Onset timing was calculated from the glove PCA data as well, seen in Figure 3-17. Using the Kolmogorov-Smirnov test, we were once again able to reject the null hypothesis that the two datasets are from the same distribution. This suggests that in this set of samples, subjects are starting grip formation at different times for power and precision grips. As with the accelerometer z-axis onset timing calculations, this assumption depends on the conditions used for determining the onset timing.

Applying *k*-means clustering to the glove principal component 1 signals did not return as high accuracy percentages as with the wrist rotation angles. Only 68.1% of objects in the power grip category were classified into the power grip cluster and 83.4% of precision grip objects. Since the difference between the two categories is high, many power grip trials were categorized as precision grip. This can be a sign that many of the first principal component time series signals for power grip were not as differentiable

from precision grip as previously thought. The other possibility is that subjects were forming the grip opposite of what was intended for that object image. For power and precision grips, as opposed to pronated and neutral positions, the likelihood of subjects performing the opposite behavior is more likely. Handgrips for using objects can be more subjective than hand orientations since the object image is rotated for different orientations. The method of *k*-means clustering on the first principal component may not be an appropriate one for characterizing power and precision grips from the glove data.

While these results provide valuable information about the functionality of the separate components, there are certain capabilities that were not measured in this particular test that can still provide useful information. For example, our test separated handgrips into two general groups, power and precision. The gloves record the entire range of finger flexion and extension, and can be used to characterize other grips that are in between power and precision. Additionally, for wrist rotation, our test focused on the difference between the pronated and neutral positions, which were measured by roll angle. Pitch angle was also calculated, but since the test did not incorporate movements that required various degrees of hand tilt, it was not a metric that we analyzed further. Collecting data while running different tests can provide further types of hand movement quantification.

The long-term application for this project is to measure changes in handgrip over time during rehab, for patients who have hand impairments due to stroke. The preliminary results indicate that this can be carried out. Finger extension and flexion along with wrist rotation were quantified during different handgrips and orientations. We

can feasibly implement the glove and accelerometer in a clinical environment to collect data from patients.

CHAPTER FOUR: ACQUIRE MOVEMENT DATA DURING IMAGING

4.1 Introduction

In the second part of the project, we aimed to take the hand movement tracking system a step further and demonstrate compatibility with functional neuroimaging. In addition to the accelerometer and glove data recordings outlined in the previous chapter, we also obtained MEG data representing neural activity. The same behavioral IP test from the previous chapter was administered to the subject during imaging. The glove interface code and BraviShell test code remained the same. The accelerometer was connected directly to the MEG machine. The test, data collection, and MEG was run on one healthy subject (section 3.2.3). The subject's data was analyzed and hand movements and neural activity are described below.

An overview of the system is shown in Figure 4-1. The entire data collection for the accelerometer and test remains in Matlab. Accelerometer and MEG data were collected separately in the MEG software and later imported to Matlab. All the analysis was conducted in Matlab. Much of the same analyses from the previous chapter were applied to the glove and accelerometer data obtained here. The accelerometer and glove data were once again organized into time series by trial. Both sets of conditions – power vs precision and horizontal vs vertical – were studied. For MEG data, the Matlab software package Brainstorm was used for processing and analysis.

4.2 Methods and Materials

Functional neuroimaging was incorporated into the experimental setup of the behavioral study discussed in the previous chapter. While most of the parameters

remained the same, we modified some of them, as necessary for the MEG data acquisition and analysis. Figure 4-1 illustrates the different components and steps in preparing the hand movement tracking with functional neuroimaging. It is similar to Figure 3-1, with a few exceptions, which will be explained. As in Figure 3-1, here too the blue boxes represent the external tools needed to implement into the system, the yellow boxes are C/C++ code, and the orange boxes are processes that were carried out in Matlab. The additional green boxes here represent data acquired by the MEG machine and software. A red outline indicates that the components that were developed specifically for this project.

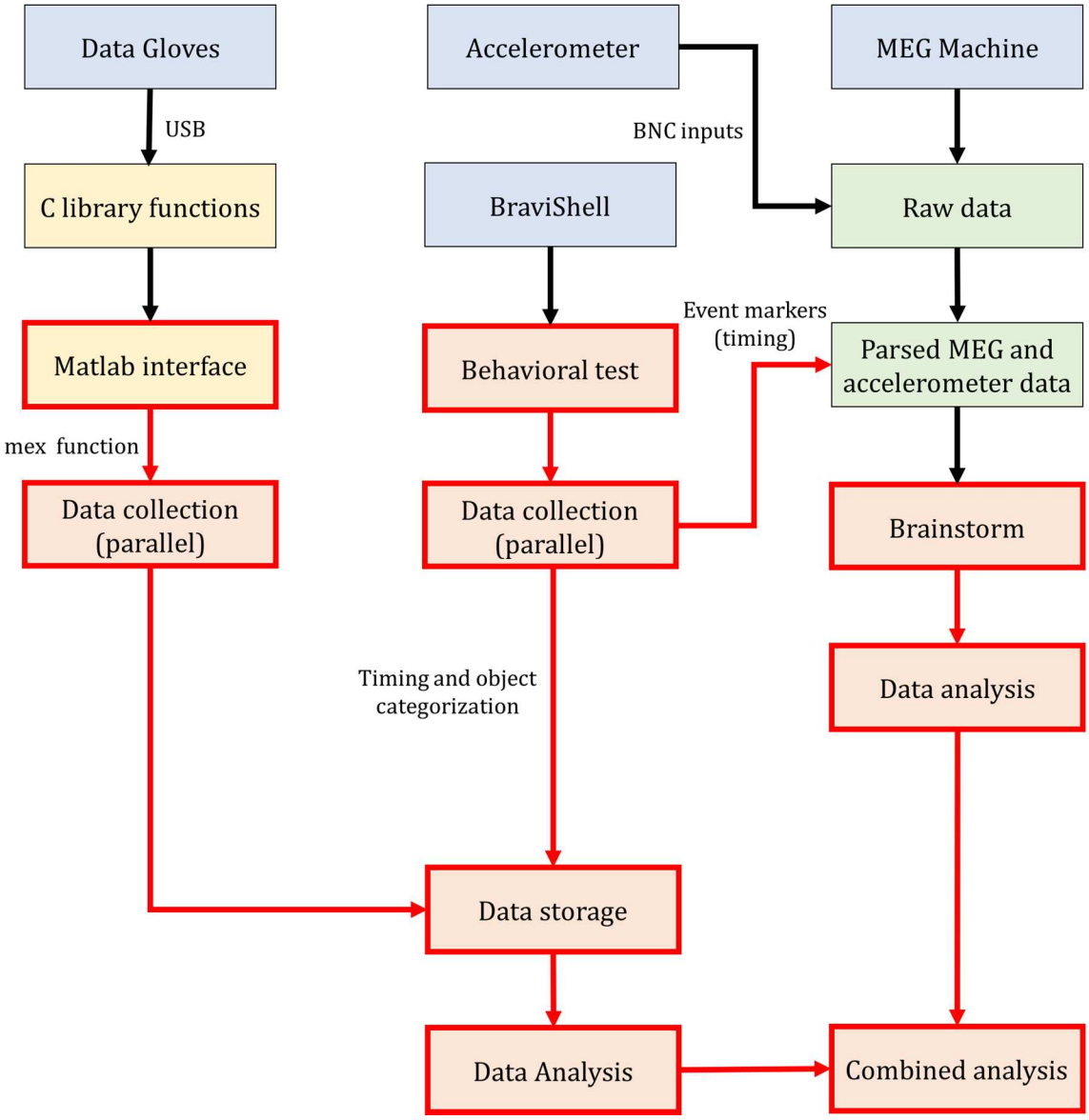

Figure 4-1: Flowchart illustrating parts of the code written for developing the hand movement tracking system combined with functional neuroimaging. The blue boxes represent the separate external components of this system, the yellow boxes represent C/C++ code including the hardware SDK and wrapper code, the green boxes represent data acquired by the MEG machine, and the orange boxes represent Matlab functions and scripts for reading, recording, and saving data. Arrows and boxes outlined in red were developed from scratch for this system. Descriptions adjacent to arrows describe the method used or information being transmitted at those steps.

4.2.1 Accelerometer and data glove

No changes were made to the glove data collection scripts used in the behavioral-only study (section 3.2.2). It was run in the same manner as in the previous chapter, parallel to the IP test and MEG scanning. The glove code did not have any direct interaction with the MEG equipment. During the scanning session, the gloves were taken into the magnetically shielded MEG room through an opening that allows for external connections during scanning.

The accelerometer was directly interfaced with the MEG machine, and therefore the DAQ and the interface code were not necessary. The three accelerometer BNC outputs were connected to BNC inputs on the MEG equipment, to record data directly alongside the MEG signals. No further programming was necessary. The entire accelerometer unit was brought into the MEG room, with the BNC connections outside the room. As in the behavioral tests, the accelerometer was taped to the back of the glove.

4.2.2 Subject

One healthy right-handed volunteer (male, age 29), with no known neurological or psychiatric disorders, participated in an MEG scanning session while the IP test was administered and hand movement data collected. There were seven blocks in a single session, over the course of 70 minutes, including sufficient breaks throughout. The subject was familiar with the instructions and had practice trials before arriving for the scanning. Informed consent according to the Declaration of Helsinki (2008) was obtained from the subject. The structural MRI data, necessary for mapping MEG output data, was collected during a separate scanning session.

4.2.3 Behavioral test

The same IP test used in the behavioral-only study (section 3.2.4) was administered to the subject. The test paradigm illustrated in Figure 3-5 is not only a means of collecting behavioral data, but was originally designed in such a way that meaningful MEG signals could be obtained. For example, the 500 milliseconds Fixation phase is the baseline period used as the reference period for mapping cortical activation from the MEG data. Additionally, the IP test paradigm separates the visual stimulus and the handgrip formation phases – the visual stimulus appears on-screen for 1500 milliseconds, followed by the cue to the subject to form the handgrip, which also lasts for 1500 milliseconds. This was done in order to differentiate the neural processes in each step. The object recognition and motor planning occur in the first phase, while the image is presented and the motor action is limited to the second phase, during the "form grip" cue screen. The trials in the test were designed in this way in order to isolate the handgrip motion in its own time window, which simplifies the analysis.

Another consideration in the IP test paradigm for collecting meaningful MEG data was that subjects respond to visual stimuli instead of grasping physical objects. This would reduce somatosensory activation signals in the cortex, making other relevant activation (such as in the motor cortex) less noisy. Moreover, both natural and man-made objects were tested to understand the varying effects of each type of object. It has been reported that recognition of man-made objects depends on functional features, such as what the object would be used for, while recognition of natural objects relies more on visual features such as color or shape (Carlson and Van Der Zee, 2004; Warrington and

Shallice, 1984). We hypothesize that subjects would make these considerations not just for identifying the object, but also for forming the appropriate grip appropriate for grasping it.

4.2.4 MEG data acquisition

The MEG study was conducted at the Athinoula A. Martinos Imaging Center at the Massachusetts Institute of Technology (Cambridge, Massachusetts). The subject was seated in a chair under the MEG sensor array and faced the projection screen placed at a distance of 80 cm. MEG data were acquired at 1000 Hz with a 306-channel Neuromag Vectorview whole-head system (Elekta Neuromag Finland) comprising 204 orthogonally oriented planar gradiometers and 102 magnetometers at 102 locations. The entire system was housed in a magnetically shielded and sound-proof room (Imedco AG, Switzerland). To compute the head position inside the MEG, four head-position indicator (HPI) coils were fixed on the subject's head. The positions of the HPI electrodes on the head as well as at least 80 points on the scalp were entered with a magnetic digitizer (Polhemus Fastrak 3D) in a head coordinate frame defined by the nasion and the left and right auricular points (fiducial landmarks). The MEG signals were band-pass filtered to the frequency range 0.5 – 200 Hz and digitized at 600 Hz. The stimuli were projected onto a 44” diameter screen through an aperture in the MEG chamber using a Panasonic DLP projector (Model #PT-D7500U) at a resolution of 1024x768 pixels with a 75 Hz refresh rate. During the experiment the room lighting was dimmed.

On a separate day than the MEG study, a T1 weighted structural MRI scan was acquired for the subject using an 8 channel phase array head coil in a 3T scanner

(Siemens-Trio, Erlagen, Germany) at the Athinoula A. Martinos Imaging Center at the Massachusetts Institute of Technology. Parameters of the sequence were; distance factor: 50%; slices per slab: 128; FOV: 256; FOV phase: 100; slice thickness: 1.33mm, TR: 2530ms, TE: 3.39ms. The cortical map and volumetric segmentation of structural MRI data was performed on Freesurfer (http://surfer.nmr.mgh.harvard.edu; Fischl et al., 2002; Fischl et al., 2004; Dale et al., 1999; Fischl et al., 1999). Additionally, MNE software (http://www.nmr.mgh.harvard.edu/martinos/userInfo/data/sofMNE.php; Gramfort et al., 2014) was used for aligning the subject's MEG data onto the structural MRI of the brain.

Besides the accelerometer data acquisition, the other external variable within the MEG data are the event markers, indicating a visual stimulus. Every time a stimulus appears in the IP test (when an object image is presented to the subject) the time point is saved in a separate file with the MEG signals. These event markers throughout the course of the MEG scan are necessary for meaningful post-processing and data analysis. They are used as the reference point for specifying the time epochs per trial. Additionally, a uniform time scale was set in order to compare data between the gloves and the MEG scanning. The event marker time points in the MEG data correspond to the visual stimulus time points imported into the glove data from the IP test output.

4.2.5 Data analysis

The glove data was saved in the same manner as in the behavioral-only study (section 3.2.6). We also performed the same analysis as in the behavioral-only study. After resampling and organizing the time series data into per trial data, they were categorized into power and precision. PCA was applied to the 14-sensor data and the first

principal component trial averages were plotted. The accelerometer data was also analyzed in the same way as in the behavioral-only study (section 3.2.6). We looked at the raw data, calculated onset timing, measured roll and pitch angles, and performed *k*-means clustering. However, since the accelerometer was directly interfaced with the MEG machine, the difference was in importing and organizing the data to the Matlab workspace from Brainstorm. In Brainstorm, the accelerometer data was separated by each of the seven blocks, rather than being one continuous time series like in the behavioral-only study. Additionally, since the sampling rate was much higher, at 1000 Hz, a moving average filter with a time window of 20 milliseconds was applied to reduce the noise.

The primary interfacing that had to be done between the data glove and MEG scanning was synchronizing the timing for data analysis. This was achieved with the timing recorded by the IP test, which was common between the two components. The IP test created event marker time points in the MEG data. In this case, event markers were placed at the visual stimulus presentation, which we used as the reference for the time epoch per trial. Since the IP test also included object category information, the event markers in the MEG data were categorized into each of the four object type categories (power natural, power man-made, precision natural, and precision man-made) and separately into each of the two orientation categories (horizontal and vertical), making various group comparisons possible.

Since the behavioral test code and the glove code are run separately, they do not share the same timescale. However, each code can individually access the machine's internal clock. This shared timescale is then used to incorporate the same event markers

as the MEG data in the glove data. As was done in the behavioral-only study (section 3.2.6), the event markers in the glove data were used to create averaged trial data by conditions.

The MEG data was imported into Brainstorm (Tadel et al., 2011), a Matlab software package used for MEG and EEG data processing and analysis. The following pre-processing steps were done after importing the data: registering the anatomical MRI data, removing line noise and eye blinks from the signals, and rejecting bad trials. Bad trials were determined by visual inspection of the MEG data. Time segments of the signal that were considered significantly noisy or contained artifacts were marked as bad. Any trials that were within these segments were rejected. For each trial, the MEG signals were separated into 4 second time epochs, starting at 1 second prior to the visual stimulus presentation event marker to 3 seconds after. In order to visualize cortical activation more clearly, Brainstorm's spatial smoothing and scaling features were applied to the cortical map.

4.3 Results

4.3.1 Raw accelerometer data

After importing the raw accelerometer data into Brainstorm and saving in Matlab, the data was organized by blocks and trials, and filtered. Figure 4-2 shows an example of the raw accelerometer data from the three axes for a single block, or 104 trials. The x-axis is the time in seconds and the y-axis is the output voltage, representing acceleration.

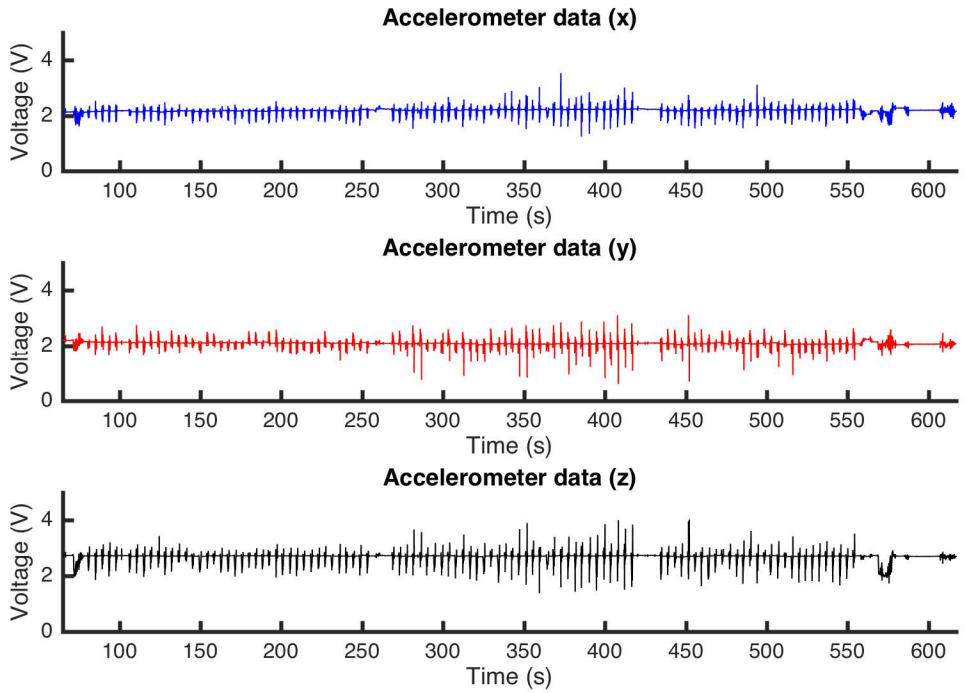

Figure 4-2: Raw accelerometer time series data from three channels (x, y, and z axes) during one IP test. The x-axis in each plot represents the time in seconds during the course of one block. The y-axis in each plot represents the voltage output reading from the respective axis of the accelerometer.

From the three axes raw accelerometer data, the z-axis represents the most useful information, since the subject lifted his hand along the z-axis from resting to forming the grip in each trial. Figure 4-3 shows the single trial average for the z-axis, with standard error. The time series plotted is an average of 364 trials in each of the power and precision categories. The timing is the same as the figures in previous chapters, with -1.5 to 0 seconds during the visual stimulus presentation and 0 to 1.5 seconds during the grip formation phase.

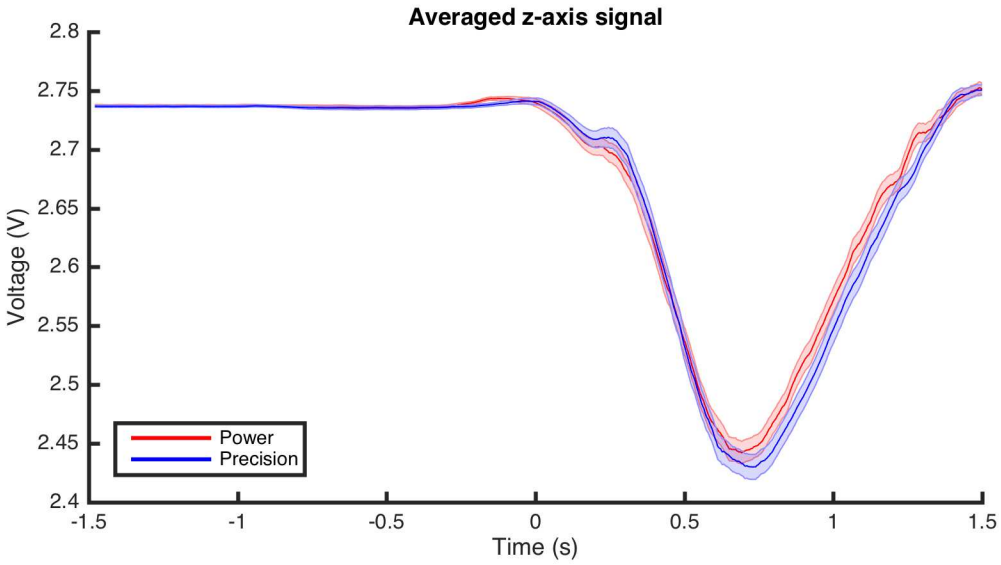

Figure 4-3: Average z-axis accelerometer data during one trial, categorized into power and precision grip trials. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the voltage reading. This plot illustrates the reading from only the z-axis channel on the accelerometer. The red line represents the mean for the trials categorized as power grip (n=364) and the blue line represents the mean for the trials categorized as precision grip (n=364). Both lines contain shaded areas representing their respective standard error.

The primary metric we obtained from the z-axis accelerometer data is the onset timing. As described in further detail previously, this is the amount of time that passes after the grip formation cue has appeared on the screen until the subject started moving his hand along the z-axis to start forming the grip. The distribution of onset times for power (362 trials) and precision (363 trials) categories are shown in Figure 4-4. The mean onset time for power grip is 0.32 seconds (SD = 0.20) and the mean onset for precision grip is 0.34 seconds (SD = 0.20). A Kolmogorov-Smirnov comparison test showed no statistical difference in the two distributions (p-value of 0.43).

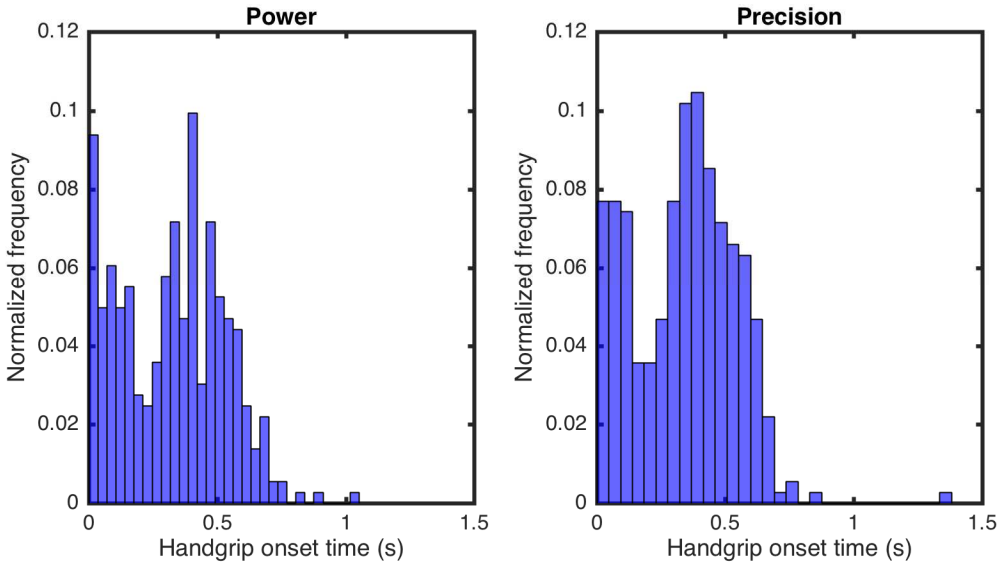

Figure 4-4: Distribution of onset times across all trials, as calculated from z-axis acceleration, for power and precision grip trials. The x-axis in each histogram shows the amount of time in seconds after the "Form grip" cue is presented to the subject. The y-axis in each histogram is the proportion of samples from the total sample size for the respective category. The power grip trials (histogram on left) has a sample size of 362 and the precision grip trials (histogram on right) has a sample size of 363. The onset time represents the amount time, after being cued, that it takes the subject to start making the handgrip response. Both distributions are centered around approximately 0.45 seconds, with a peak near 0 seconds as well.

4.3.2 Angle of rotation

As we did with the behavioral test data in the previous chapter, we also found the roll and pitch angles from the raw accelerometer data for this subject. These measurements are useful in telling us about the angular displacement of the hand and wrist in two degrees of freedom. The 364 trials in each of the pronated and neutral position categories were averaged and the results shown in Figure 4-5. The x-axis time scale is the same as Figure 4-3 and the y-axis the angle of rotation, with increasing values representing clockwise rotation. The mean baseline angle for each trial was subtracted from the total time series in order to set the resting position to 0 degrees, normalizing the angular displacement across trials.

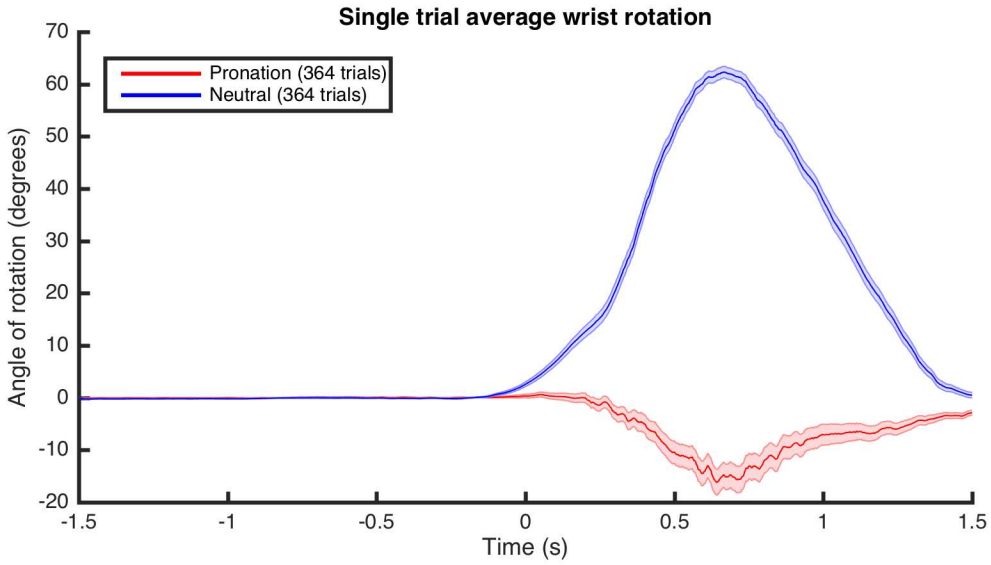

Figure 4-5: Average roll angle during one trial, categorized into pronation and neutral position trials. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the roll angle measurement (which we are using as a correlate for wrist rotation) in degrees. A clockwise rotation is represented by an increase in angle. The red line is the mean of all the trials categorized as pronation position (n=364) and the blue line is the mean of all the trials categorized as neutral position (n=364). Both lines contain shaded areas representing their respective standard error.

For each of the 364 trials in the two categories, we determined the maximum angular displacement. Figure 4-6 is a distribution of the maximum angular displacement (positive or negative) in each trial, grouped by pronation and neutral conditions. The response to objects categorized as pronation was mostly a small counterclockwise wrist rotation, while the response to neutral was typically a larger clockwise rotation. The mean maximum angular displacement was -47.9 degrees (SD = 71.76) for the pronation group and 78.81 degrees (SD = 29.9) for the neutral group.

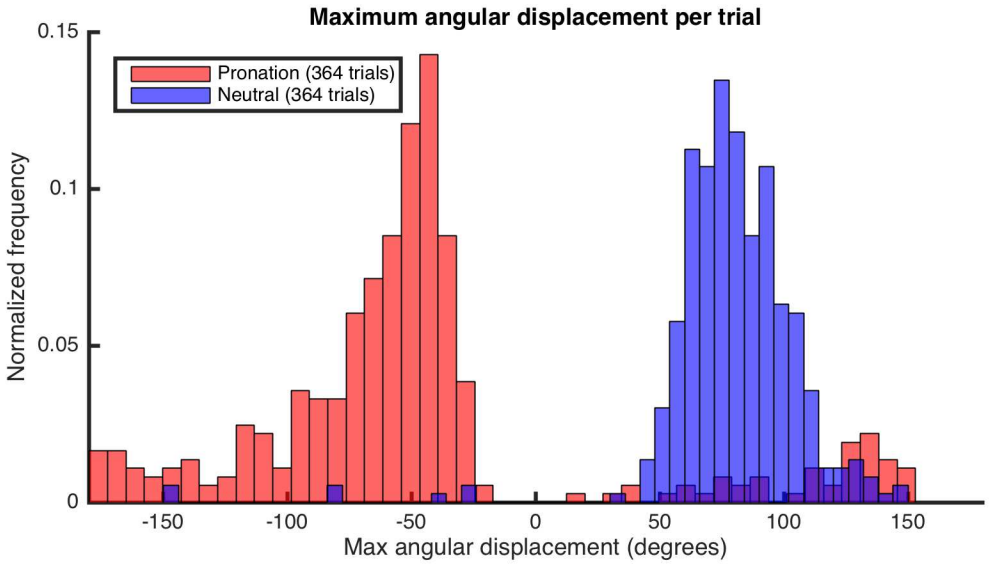

Figure 4-6: Distribution of maximum angular displacement across all trials, calculated from roll angle, categorized into pronation and neutral position trials. The x-axis shows the angular displacement in degrees. Maximum angular displacement is calculated as the difference between mean roll angle at resting position and maximum absolute value of roll angle during grip formation, for each trial. This represents the greatest magnitude of wrist rotation (clockwise or counterclockwise) during each trial. The y-axis shows the proportion of samples from the total sample size for the respective category. The distribution of pronation position trials (in red, n=364) has a peak at approximately -50 degrees and the distribution of neutral position trials (in blue, n=364) is centered around approximately 75 degrees.

We attempted *k*-means clustering on the angular displacement time series for all 728 trials to try to classify the data into our two desired groups. Out of the 364 trials originally assigned to the pronation group, 88.2% were correctly clustered. This value was 98.4% for the neutral group. In order to more clearly visualize these results, the same data from Figure 4-5 was plotted in 4-7, except the grouping was done by the *k*-means clustering results.

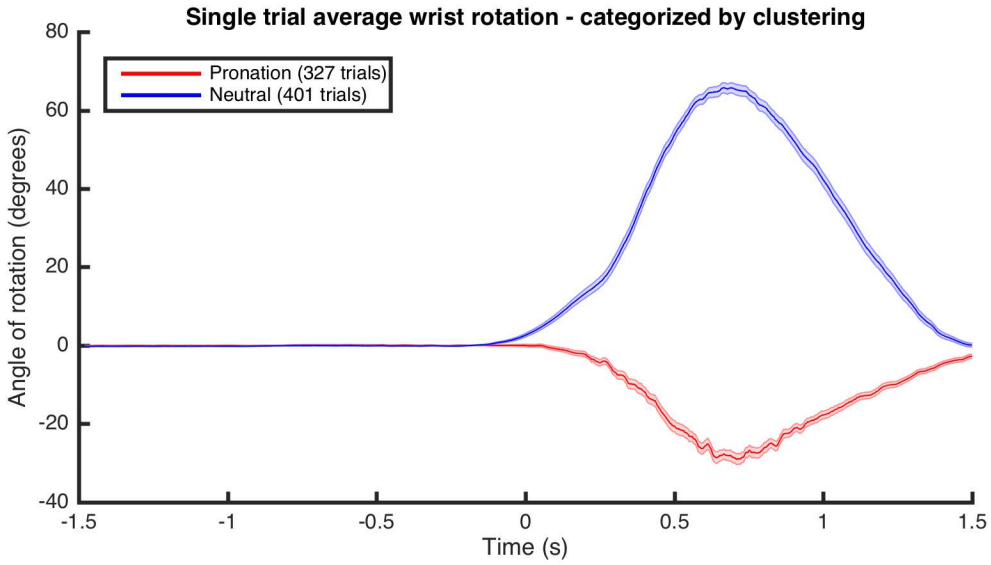

Figure 4-7: Average roll angle during one trial, categorized into pronation and neutral position trials based on clustering results. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the roll angle measurement (which we are using as a correlate for wrist rotation) in degrees. A clockwise rotation is represented by an increase in angle. The red line is the mean of all the trials categorized as pronation position (n=327) and the blue line is the mean of all the trials categorized as neutral position (n=401). Both lines contain shaded areas representing their respective standard error.

4.3.3 Glove data

The glove data was saved and analyzed in the same way in this part as in the previous chapter. Figure 4-8 is an example of the results obtained from all 14 sensors during a single block within the entire scan session. 104 trials are represented in this time period, and each vertical red line represents the visual stimulus appearing on the screen. The axes and values are the same as described for Figure 3-14.

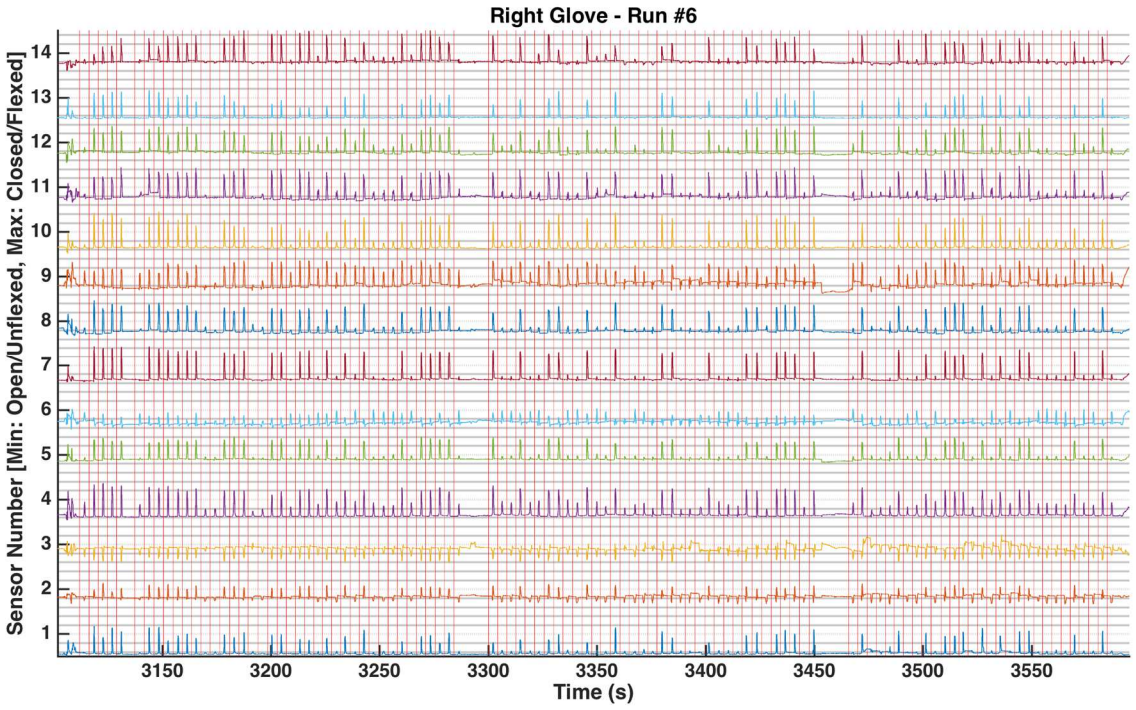

Figure 4-8: Data glove recordings from all 14 sensors during one IP test block. The x-axis represents the time in seconds during the course of one block. Time does not start at 0 seconds since this is one block within a series of blocks. The y-axis contains 14 separate time series plots, each representing an individual glove sensor. The sensors are located on the along the back of the glove, with two flexion sensors on two joints at each finger and one abduction sensor between each finger. Sensors are numbered 1 to 14 in order from left to right, starting at the base of the thumb and ending at the second joint of the pinky. For each sensor, the output value is between 0 and 1, with higher values representing flexion or adduction, depending on sensor location. In certain sensors, differences are visible based on grip type. The vertical red lines are the stimulus markers, indicating the time at which the visual stimulus was presented to the subject on the screen.

Once again, PCA was applied to the 14-sensor data in order to reduce the dimensionality and produce a single time series describing the total handgrip movement. Figure 4-9 is a time series for the first principal component from the results of conducting PCA on the data from Figure 4-8.

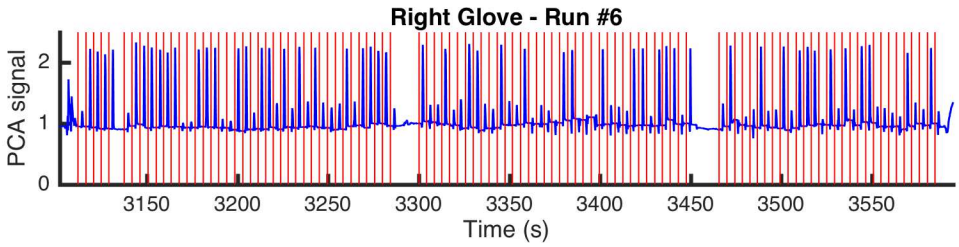

Figure 4-9: Principal component 1 during one block, calculated from 14 sensor glove data. The x-axis represents the time in seconds during the course of one block. Time does not start at 0 seconds since this is one block within a series of blocks. The y-axis shows the magnitude of flexion or adduction, with increasing values representing further finger flexion or adduction. Since this is described by a principal component, the y-axis value is a combination of the flexion and adduction sensor outputs, and does not completely represent one or the other. The vertical red lines are the stimulus markers, indicating the time at which the visual stimulus was presented to the subject on the screen.

As pictured in Figure 4-9, the timing for each visual stimulus was saved, and was used to organize the PCA data into trials by power and precision categories. The average time series for each trial by category is plotted in Figure 4-10. 364 trials were averaged for each category. The amplitude of the signal generated from the power grip objects is greater than that of the signal from the precision grip objects. The time along the x-axis remains the same as in previous figures, representing the visual stimulus phase followed by the grip formation phase.

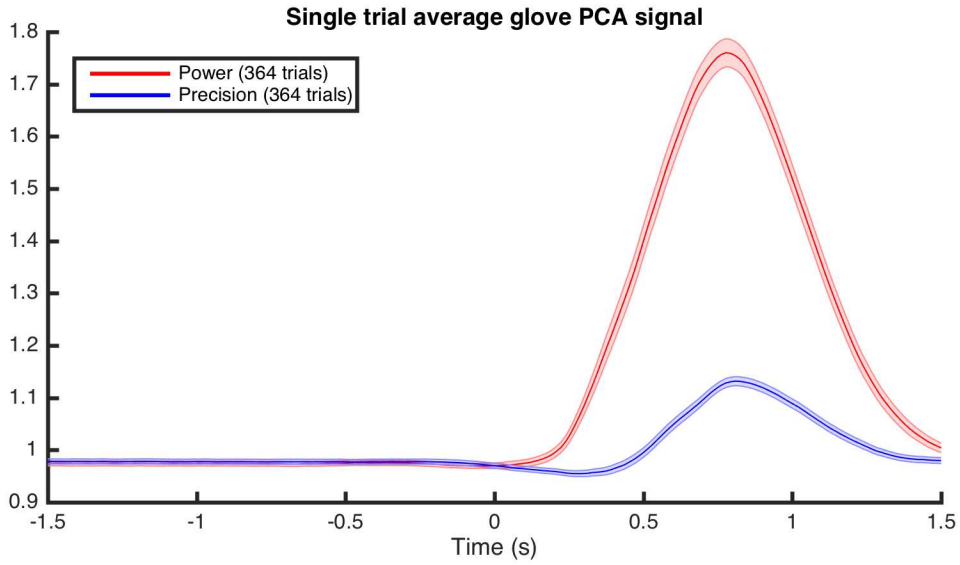

Figure 4-10: Average principal component 1 signal during one trial, calculated from 14 sensor glove data, categorized into power and precision grip trials. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the magnitude of flexion or adduction, with increasing values representing further finger flexion or adduction. Since this is described by a principal component, the y-axis value is a combination of the flexion and adduction sensor outputs, and does not completely represent one or the other. The red line is the mean of all the trials categorized as power grip (n=364) and the blue line is the mean of all the trials categorized as precision grip (n=364). Both lines contain shaded areas representing their respective standard error.

Onset timing was calculated for grip formation using the first principal component data from each trial. The distribution of times is seen in Figure 4-11. The first principal component signals provide a helpful correlate for finger flexion, indicating the time course of the grip being formed. As in previous histogram figures, the time at 0 seconds in Figure 4-11 is the start of the "form grip" cue appearing on the screen. Onset timing in this instance measures the amount of time between the appearance of that cue and the beginning of the grip being formed. The power grip category consists of 357 trials and the precision grip category 344 trials. Out of 364 trials for each, the ones

excluded most likely did not reach the designated threshold for calculating onset timing. The mean onset time for the power grip object images was 0.46 seconds (SD = 0.25) and the onset time for precision grip was 0.54 seconds (SD = 0.22). A Kolmogorov-Smirnov comparison test was run between the two sets of data, concluding that there is a statistical difference (p-value of 7.60e-06).

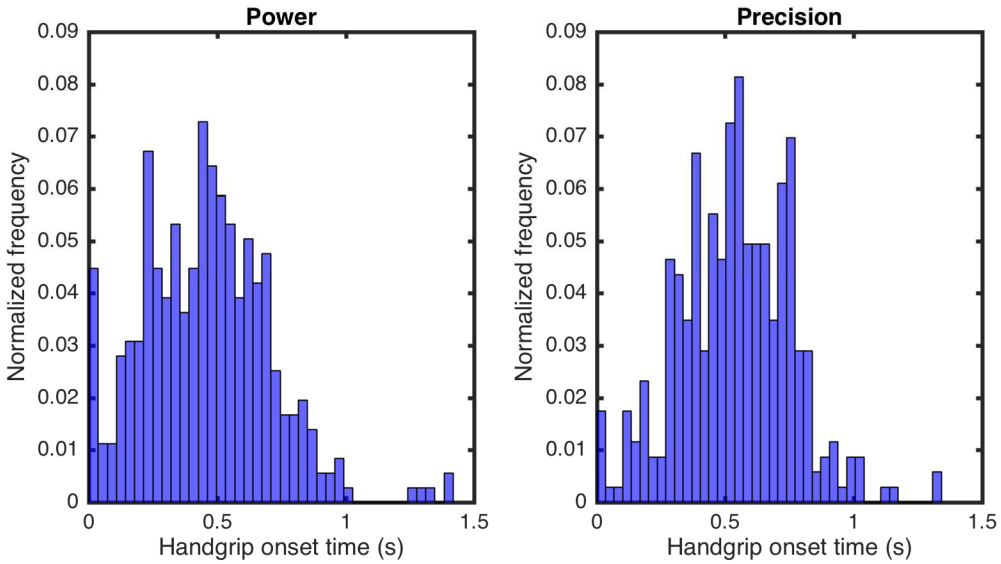

Figure 4-11: Distribution of onset times across all trials, as calculated from data glove principal component 1 signal, for power and precision grip trials. The x-axis in each histogram shows the amount of time in seconds after the "Form grip" cue is presented to the subject. The y-axis in each histogram is the proportion of samples from the total sample size for the respective category. The power grip trials (histogram on left) has a sample size of 357 and the precision grip trials (histogram on right) has a sample size of 344. The onset time represents the amount time, after being cued, that it takes the subject to start making the handgrip response. Both distributions are centered around approximately 0.50 seconds, with a peak near 0 seconds as well in the power grip trials.

We applied *k*-means clustering once again to classify the glove's principal component 1 signals into power and precision grip trials. Out of the 364 trials categorized as power grip, 74.2% were in the power grip cluster. For the 364 trials of precision grip, 99.7% were correctly identified as precision grip trials. Figure 4-12 is a visualization of

the clustering outcome. The per trial average for all 728 first principal component time series were plotted, and grouped by the clustering results.

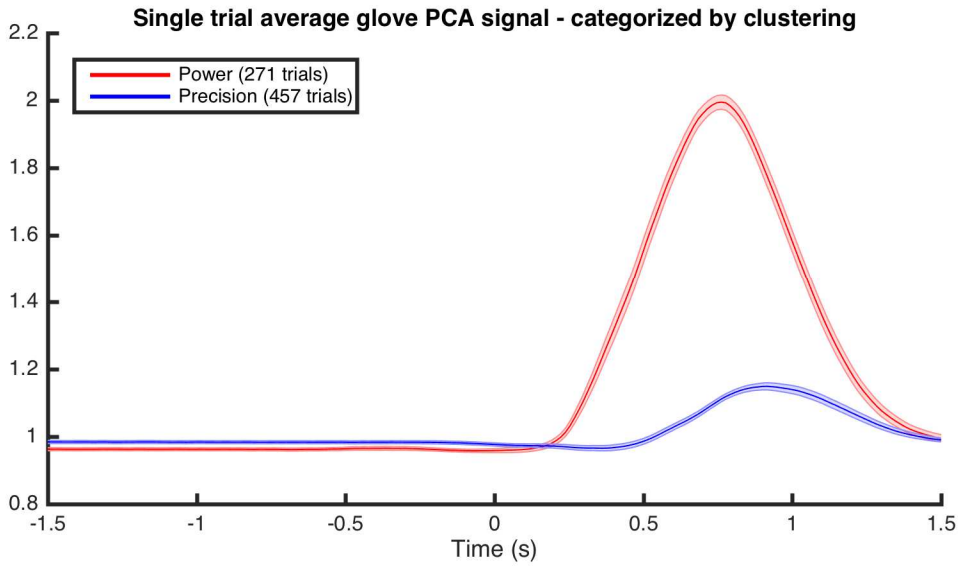

Figure 4-12: Average principal component 1 signal during one trial, calculated from 14 sensor glove data, categorized into power and precision grip trials based on clustering results. The x-axis shows the time during a single trial, with 0 seconds representing the time at which the subject starts making the handgrip response. -1.5 to 0 seconds is the visual stimulus presented to the subject (hand is in resting position) and 0 to 1.5 seconds is the "Form grip" text presented to the subject (handgrip is being formed). The y-axis shows the magnitude of flexion or adduction, with increasing values representing further finger flexion or adduction. Since this is described by a principal component, the y-axis value is a combination of the flexion and adduction sensor outputs, and does not completely represent one or the other. The red line is the mean of all the trials categorized as power grip (n=271) and the blue line is the mean of all the trials categorized as precision grip (n=457). Both lines contain shaded areas representing their respective standard error.

4.3.4 MEG data

The MEG data processing and analysis was primarily carried out on Brainstorm in Matlab. The signals from each of the seven blocks were imported to Brainstorm separately as time series. Each trial was represented by a 4-second time epoch and the trials were separated into the four object categories (natural power grip, man-made power grip, natural precision grip, and man-made precision grip). Averaged cortical maps over

time for each category were generated. Spatial smoothing and amplitude scaling were used to more clearly map the activation. The averaged activation across the cortex over time for each group was reviewed to determine particular regions of interest. The FreeSurfer surface-based Destrieux atlas (Destrieux et al., 2010) was used to identify cortical regions. We focused on two distinct time points during the course of the trial, selected based on visual inspection of activation over time. The significance of the active regions is explained for each time point.

Figure 4-13 shows the averaged cortical activation for the four categories (power man-made: 178 trials, power natural: 176 trials, precision man-made: 174 trials, precision natural: 179 trials) at 273 milliseconds after the visual stimulus appeared on the screen. (This time corresponds to -1.26 seconds in the timescale seen in the single trial average figures from sections 4.3.1, 4.3.2, and 4.3.3.) This is the time point at which significant activation started appearing in the visual cortex. The colorbar represents the activation magnitude as a normalized version of the current [A-m] found in each point. The normalization method implemented is dynamical Statistical Parametric Mapping (dSPM) (Dale, 2000), which uses noise variance estimates at each point to normalize the current map by standard deviation of the variance estimates. This gives a unitless z-score statistical map, which is what the scale on the colorbar represents. During this time, the object image has been presented and the subject is focused on recognizing the object. Generally speaking, we see activation in parts of the visual cortex and lateral surface of the parietal lobe at this time point.

Several sulci and gyri are highlighted. The regions around the visual cortex will be described first. The regions outlined in navy blue are the *superior occipital* and *transverse occipital sulci*, grouped together, in both the left and right hemispheres. The region in green adjacent to it is the *left superior occipital gyrus*. The region outlined in red, pictured in the right hemisphere in the figure, is the *cuneus*. These three regions are all part of the extrastriate cortex, which is a visual association area with functions including feature extraction, shape recognition, and attention. Since the subject is viewing the object image at this time, activation in these areas are expected.

There is another significant area of activation in the figure, pictured on the left hemisphere in the anterior direction. The region outlined in purple is the *intraparietal sulcus* and *transverse parietal sulcus*, grouped together, and the region in blue adjacent to it is the *angular gyrus*. The intraparietal sulcus plays a role in visual control of grasping and manipulating hand movements. Since the subject is planning a grip for the object being presented, activation in this area is relevant. The areas of activation at this time point seem to be correlated, as they tend to increase and decrease together during other time points in the trial.

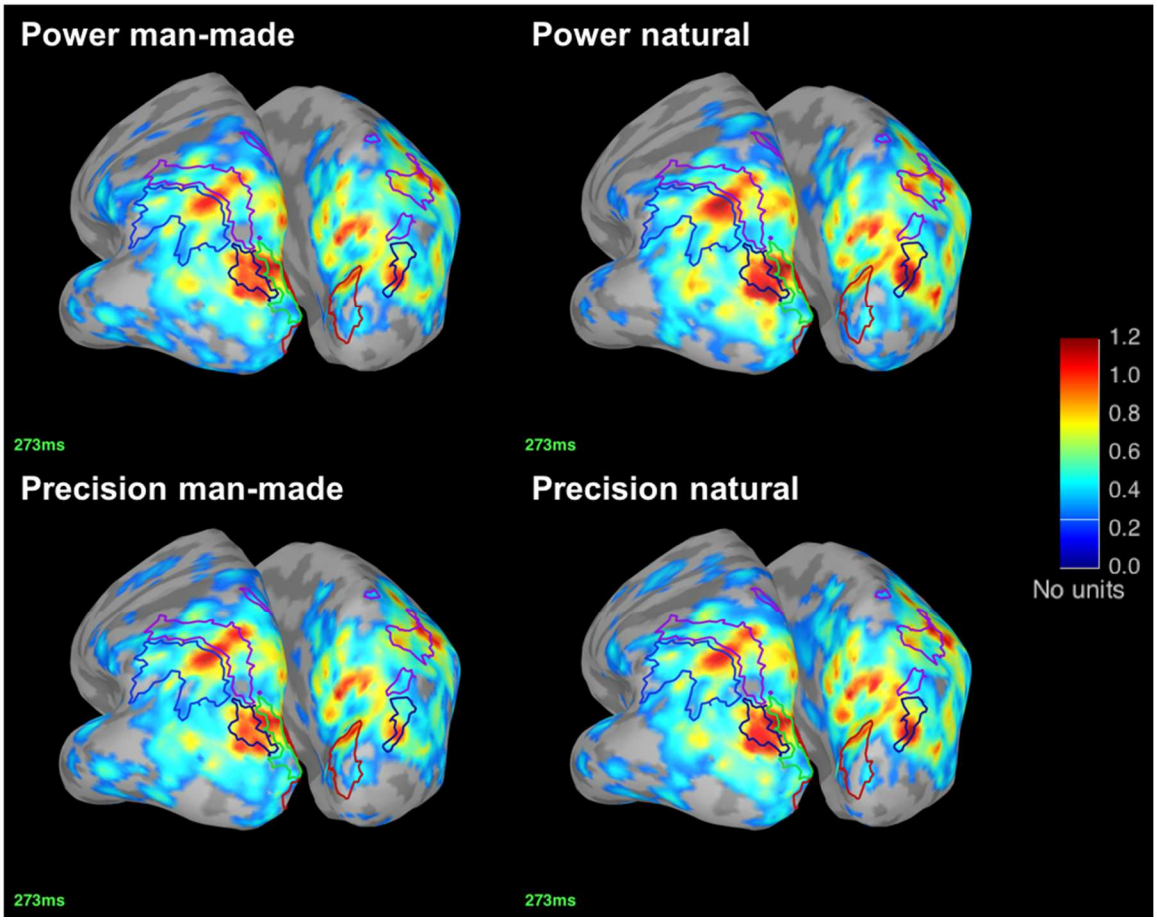

Figure 4-13: Average cortical activation at 273 ms after object image is presented, for each of the four object types (power man-made: 178 trials, power natural: 176 trials, precision man-made: 174 trials, precision natural: 179 trials). The colorbar is a z-score, representing the current at each point normalized by standard deviation of the noise variance at each point. There is significant activity around the visual cortex (navy blue = superior occipital and transverse occipital sulci, green = left superior occipital gyrus, red = cuneus) and in the lateral surface of the parietal lobe (purple = intraparietal sulcus and transverse parietal sulcus, blue = angular gyrus). Activation in both these areas are correlated with different functions of visual processing. Additionally, these areas tend to exhibit similar activation patterns over time during the course of the trial.

Figure 4-14 shows the activation maps, at 1890 milliseconds after the visual stimulus appeared on the screen, or 390 milliseconds after the "form grip" cue appeared. (This corresponds to 0.39 seconds in the timescale seen in the single trial average figures from sections 4.3.1, 4.3.2, and 4.3.3.) At this time, the subject is about to or has started

moving his hand to form the necessary grip. This time point was selected since there is significant activation in the left primary motor cortex. The colorbar is the same z-score described for Figure 4-13. However, the scaling is different, as the maximum z-score is set higher to 1.5 rather than 1.2. This is done in order to visualize the activation more clearly, but since the scaling is different between figures, they cannot be directly compared. There is activation at the motor cortex, as well as some areas in the anterior parts of both the frontal and temporal lobes.

The first area with significant activation is the motor cortex. The region outlined in blue is the *precentral sulcus*, the adjacent region in gray is the *precentral gyrus*, and the region adjacent to that in red is the *central sulcus*. These comprise the primary motor cortex, which is responsible in part for executing motor functions. The particular section of the motor cortex where there is activation represents the hand and finger, since the subject is starting to form the handgrip around this time.

The second area with activation is at the anterior part of the frontal lobe. The region outlined in light yellow is the *triangular part of the inferior frontal gyrus*, the region adjacent to it in light red is the *orbital gyrus*, and the region outlined in dark red adjacent to it is the *H-shaped orbital sulcus*. These parts of the brain are part of the orbitofrontal cortex, which hypothesized to be involved in the process of decision-making. Activity in this region may indicate that the subject is determining which grip to form.

Lastly, there is also activation in the anterior part of the temporal lobe. The region outlined in red is the *middle temporal gyrus*, the region outlined in yellow is the *superior*

temporal sulcus, and the thin region outlined in pink is the *inferior circular sulcus of the insula*. The parts of these regions that were active make up the temporal pole, which is known for semantic processing. However, this function would not be indicative of the subject's behavior at the time.

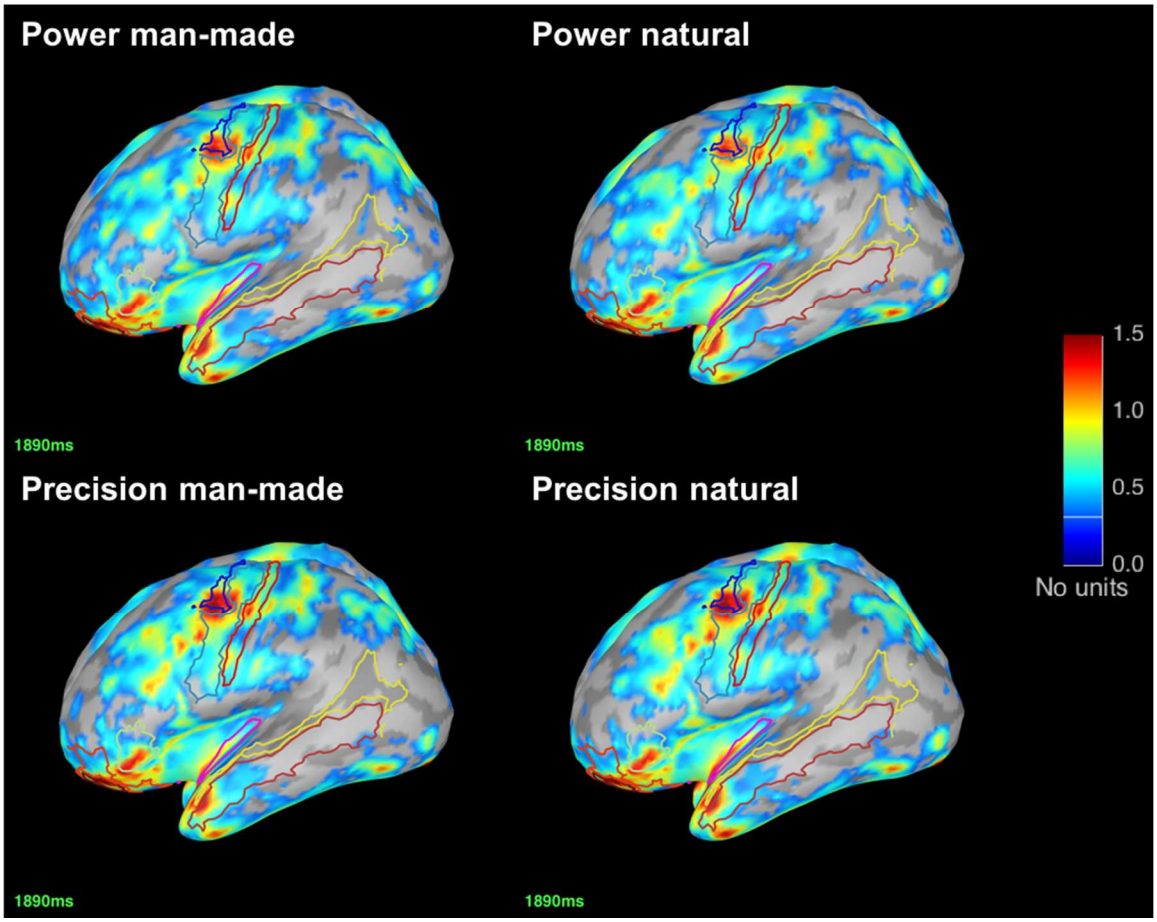

Figure 4-14: Average cortical activation at 390 ms after grip formation cue is presented, for each of the four object categories (power man-made: 178 trials, power natural: 176 trials, precision man-made: 174 trials, precision natural: 179 trials). The colorbar is a z-score, representing the current at each point normalized by standard deviation of the noise variance at each point. There is significant activity at the motor cortex (blue = precentral sulcus, gray = precentral gyrus, red = central sulcus), the orbitofrontal cortex (light yellow = triangular part of the inferior frontal gyrus, light red = orbital gyrus, dark red = H-shaped orbital sulcus), and temporal pole (red = middle temporal gyrus, yellow = superior temporal sulcus, pink = inferior circular sulcus of the insula). Activation in the hand and finger aspects of the motor cortex are evident. It is less clear the relevance of the activation in the other two areas.

4.4 Discussion

In this second part of the thesis, we conducted the behavioral IP test during a functional neuroimaging session to collect meaningful information about hand movement and neural activity. From the accelerometer data, we viewed the raw accelerometer data, categorized trials into power/precision grips and pronated/neutral positions calculated onset timing, measured roll and pitch angles, and applied *k*-means clustering. From the glove data, we viewed the 14-sensor raw glove data, applied PCA, categorized trials into power/precision grips, calculated onset timing, and applied *k*-means clustering. These analyses for both the accelerometer and data glove were performed to demonstrate that the results of of each were returning the expected results. Additionally, the results from the MEG scan session were processed using Brainstorm and the cortical activity studied.

Since the accelerometer data was recorded directly from the MEG equipment for this part, rather than through the DAQ device, it was important to ensure that the results obtained were consistent with what we expected based on previous data. The raw data was recorded at 1000 Hz (compared to 45 Hz with the DAQ), which generated significant amounts of high-frequency noise. A moving average filter with a time window of 20 milliseconds was implemented in Matlab to reduce the noise and smooth out the time series data. In the example raw data from Figure 4-2, we see that the overall magnitude of the x- and y-axis signals is slightly less than the z-axis output. This confirms that the acceleration due to gravity is being detected properly by the accelerometer.

The z-axis data was then plotted as an average by trial for the power and precision categories, in Figure 4-3. The statistical test for significance failed to reject the null

hypothesis, indicating that there is no statistical difference between the two distributions. This is unlike the comparison done for the same measurement in the previous chapter. There were fewer samples in this case and all the data was from a single subject, so there would be less variability expected in the results. Additionally, in the histogram, we see that there is a higher frequency of onset times close to 0 seconds. These were likely the result of the subject forming the grip too soon, perhaps starting the motion before the cue appeared on the screen. These values affect the comparison test of course.

The roll and pitch angles were calculated from the raw accelerometer data. In Figure 4-5, the average rotation angle time series are plotted for the subject. The outcome is aligned with what we would expect to see – high clockwise rotation during the neutral position trials and a slight counterclockwise rotation for pronated position trials. However, compared to the averaged results seen in the first part (Figure 3-11), this subject had a more sustained counterclockwise rotation from resting position for the pronated position. This may simply be a characteristic of the hand movement for this individual subject. When averaged over several subjects, the effect of such a pattern would not be as noticeable. Additionally, the total number of trials being averaged were fewer in this case.

In the maximum angular displacement histogram in Figure 4-6, each of the two groups are clearly in two separate distributions. The pronated position group is centered around approximately -40 degrees and the neutral position group around 75 degrees. These observations are consistent with the averaged trial data from Figure 4-5. The mean, -47.9 degrees, and standard deviation, 71.76 degrees, for the pronated position group are

both fairly high values. The method for determining the maximum angle in each trial may have to be reconsidered, especially for the pronated position, since these values were also unusually high in Figure 3-12 as well.

The *k*-means clustering results for this subject produced relatively high accuracy percentages, with 88.2% for the pronated position trials and 98.4% for neutral position trials. Some of the pronation trials where the subject did not make as much of a counterclockwise rotation may have been clustered into the neutral trials group. The average angular rotation plot categorized by clusters, in Figure 4-7, would support this notion, since the pronation group dips down to -30 degrees, which is greater in magnitude than the plot in Figure 4-5. Another possibility is that the subject's grip formation may not have aligned properly with the time during which the cue appeared on the screen.

The first principal component signal over the course a single block, shown in Figure 4-9, makes the distinction between power grip and precision grip after applying PCA evident. The glove data obtained from this subject were very clear. Factors that may have contributed to this would be the subject forming very different grips for power and precision and the glove sensors fitting very well around the subject's hand. In figure 4-9, there are an equal number of trials that reach the maximum amplitude as trials that stay close to the resting position amplitude. The averaged single trial first principal component time series in Figure 4-10 looks as we would expect, with an increased amplitude for power grip trials compared to precision grip trials, since there is a greater amount of finger flexion occurring in a power grip.

The grip formation onset timing distributions based on the glove data's first

principal component are shown in Figure 4-11. The statistical test indicates that there is a significant difference between the distributions for power and precision grip trials, with a very low p-value of 7.60e-06. The precision grip trials had a higher mean onset time (0.54 seconds) than the power grip trials (0.46 seconds). This analysis would indicate that the subject was slightly slower in forming a precision grip than a power grip. There are a number of assumptions made and conditions for calculating the onset timing, some of which have already been outlined, so this conclusion should be made with those considerations in mind.

The *k*-means clustering for the principal component 1 of the glove data gave higher accuracy values for this subject than for the group of subjects in the previous chapter. However, although 99.7% of the precision grip trials were correctly identified, only 74.2% of the power grip trials were correct. Once again, some of the power grip trials were clustered with the precision grip trials. In these trials, may not have fully formed the power grip or simply formed the wrong grip.

For the MEG data, we focused on regions that we expected to see activation at different time points during the course of each trial. Figure 4-13 shows significant activation around the visual cortex at 273 milliseconds after the subject has seen the object image appear on the screen, as expected. There is also activation in the intraparietal sulcus, which is likely linked to the visual cortex activation. During the course of the trial, activation around the visual cortex typically occurs at the same time as the intraparietal cortex. Although there were minor differences in activity between each of the four object categories, none of it was statistically significant.

In Figure 4-14, activity in the left primary motor cortex is shown at 390 milliseconds after the "form grip" cue appears on the screen. This time point is between the mean onset timings for the z-axis accelerometer data and the glove PCA signal. There is also activation in two other regions – the orbitofrontal cortex and the temporal pole. While the first of those regions may be related to the subject's behavioral response, the known functions of the latter do not necessarily represent any of the subject's responses.

The results obtained in this part of the project serve to further validate that the proposed system can be used for quantifying hand movements. Additionally, we were able to collect MEG data and conduct some preliminary analyses looking cortical activation at different time points for certain regions of interest, including parts of the extrastriate cortex, the intraparietal sulcus, the primary motor cortex, the orbitofrontal cortex, and the temporal pole. Note that cortical activation was not just limited to these few areas during the entire time course of the trial, but these were selected based on activation at those particular time points.

CHAPTER FIVE: CONCLUSION

In the first part of this thesis, we administered the behavioral-only IP test to 20 subjects, collecting right hand movement data from an accelerometer and data glove. The test was designed to elicit a power or precision handgrip response, with the hand rotated in either a horizontal or vertical orientation. From the accelerometer recordings, the following calculations were done: average trial z-axis signal, onset timing from z-axis signal, roll and pitch angles, average trial wrist rotation, maximum angular displacement of wrist rotation, and *k*-means clustering. Trials were categorized into power or precision grip as well as pronation or neutral position based on the visual stimuli categories. From the glove data recordings, the following calculations were done: PCA on 14-channel glove data, average trial first principal component, onset timing from first principal component, and *k*-means clustering. Trials were categorized into power or precision grip. (The gloves cannot distinguish between horizontal and vertical orientation.) The results demonstrated that the readings from the two hardware components were able to measure subjects' hand movements, and served as an example of conclusions that can be drawn from the data.

In the second part of this thesis, we acquired the same hand movement data during a functional neuroimaging session. One subject was scanned while taking the IP test. The same calculations and categorizations described above were carried out on this subject's accelerometer and glove data as well. For the MEG data, trials were categorized into power man-made, power natural, precision man-made, and precision natural and basic MEG analysis was performed. The average cortical activation was mapped for two

time points (273 ms post-visual stimulus and 1890 ms post-visual stimulus) and regions of interest were selected based on areas of activation relevant to the behavioral response at that time. For the first time point, we looked at the visual cortex and intraparietal sulcus. The visual cortex activation can be attributed to the subject viewing the object image at this time. For the second time point, we looked at the motor cortex, orbitofrontal cortex, and temporal pole. The significant motor cortex activation, particularly at the hand and finger representation, is a result of the subject beginning his handgrip formation.

Further analysis can be done on the MEG data to quantitatively determine differences in activation for each of the categories. Frequency plots can also be generated, looking at the neural oscillation activity. Additionally, functional connectivity between regions can be calculated. The work done in this thesis serves as a proof of concept for such hardware to be used in functional neuroimaging with a behavioral test. However, the capabilities of the accelerometer and data glove hardware are not limited to the results obtained from administering this particular test. Different tests can be developed and alternative methods for data processing can be applied, in order to gain other types of hand and finger movement quantifications. The next step for this project would be to look at stroke patients who are recovering use of their hands to track motor recovery and cortical plasticity associated with training in a longitudinal study. Recording MEG data before and after rehab may reveal functional changes over time, which can also help us discover neuromarkers associated with hand motor improvement.

APPENDIX

This Appendix contains the code written for:

- interfacing the LabJack U6 DAQ with Matlab (A.1 – A.5)
- recording and saving data from the LabJack U6 DAQ (A.3 – A.6)
- interfacing the 5DT Ultra 14 Data Gloves with Matlab (A.7 – A.13)
- recording and saving data from the 5DT Ultra 14 Data Gloves (A.9 – A.15)
- select functions to execute the BraviShell behavioral test (A.16 – A.19)

Note that these scripts and functions cannot operate independently without the proper drivers and software libraries, which are not included in this Appendix.

A.1: C code to interface DAQ with Matlab

```
/*

This code is for interfacing the DAQ with Matlab. Since the DAQ SDK provides C
functions, they must be made compatible with Matlab in order to acquire and save data.
Much of this code was taken from C code in the LabJack SDK to test whether data was
being acquired properly. The relevant component that we developed is the mexFunction
starting at line 520. This code does not work on its own, but rather needs to be compiled
in and called from Matlab. Once compiled (and the mex file is generated), this file does
not need to be called on directly, as Matlab functions were written (see below).

 To use, do the following in Matlab:
   1. Navigate to the directory containing the "DAQinterfacer.c" file.
   2. Compile on Matlab using the command "mex -
I/Users/psychtests/Desktop/exodriver-master/liblabjackusb/ DAQinterfacer.c u6.c
liblabjackusb-2.5.3.dylib;". *THIS IS FOR MAC*
   3. Ensure that the file "DAQinterfacer.mexmaci64" (Mac) has been created.
   4. Run any of the following available commands:
      Input            Output
      ---------------------------------------------------------------------------------------------------
      DAQinterfacer(0)Testing and development purposes.
      DAQinterfacer(1)Initializes the DAQ.
      DAQinterfacer(2)Captures data at a single point in time from first six analog inputs
on the DAQ.
```

```
      DAQinterfacer(3)Disconnects the DAQ.

 In order to call on these commands more easily and for use in Matlab scripts, functions
were created for each one as well. These functions simply run the desired DAQinterfacer
command.
 Command          Function file
 ----------------------------------
 DAQinterfacer(1)      U6Open.m
 DAQinterfacer(2)      U6Read.m
 DAQinterfacer(3)      U6Close.m


Mahtab Alam. Updated: Apr 2017.
*/



//#include "labjackusb.h"
#include "u6.h"
#include "mex.h"
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int ConfigIO_example(HANDLE hDevice);
int StreamConfig_example(HANDLE hDevice);
int StreamStart(HANDLE hDevice);
int StreamData_example(HANDLE hDevice, u6CalibrationInfo *caliInfo);
int StreamStop(HANDLE hDevice);

const uint8 NumChannels = 5;       //For this example to work proper,
SamplesPerPacket needs
                           //to be a multiple of NumChannels.
const uint8 SamplesPerPacket = 25;  //Needs to be 25 to read multiple StreamData
responses
                           //in one large packet, otherwise can be any value between
                           //1-25 for 1 StreamData response per packet.

int localID;
HANDLE hDevice;
u6CalibrationInfo caliInfo;
long error;


//Sends a ConfigIO low-level command to turn off timers/counters
int ConfigIO_example(HANDLE hDevice)
{
   uint8 sendBuff[16], recBuff[16];
```

```
    uint16 checksumTotal;
    int sendChars, recChars, i;

    sendBuff[1] = (uint8)(0xF8);  //Command byte
    sendBuff[2] = (uint8)(0x03);  //Number of data words
    sendBuff[3] = (uint8)(0x0B);  //Extended command number

    sendBuff[6] = 1;  //Writemask : Setting writemask for TimerCounterConfig (bit 0)

    sendBuff[7] = 0;  //NumberTimersEnabled : Setting to zero to disable all timers.
    sendBuff[8] = 0;  //CounterEnable: Setting bit 0 and bit 1 to zero to disable both
counters
    sendBuff[9] = 0;  //TimerCounterPinOffset

    for( i = 10; i < 16; i++ )
        sendBuff[i] = 0;   //Reserved
    extendedChecksum(sendBuff, 16);

    //Sending command to U6
    if( (sendChars = LJUSB_Write(hDevice, sendBuff, 16)) < 16 )
    {
        if( sendChars == 0 )
            printf("ConfigIO error : write failed\n");
        else
            printf("ConfigIO error : did not write all of the buffer\n");
        return -1;
    }

    //Reading response from U6
    if( (recChars = LJUSB_Read(hDevice, recBuff, 16)) < 16 )
    {
        if( recChars == 0 )
            printf("ConfigIO error : read failed\n");
        else
            printf("ConfigIO error : did not read all of the buffer\n");
        return -1;
    }

    checksumTotal = extendedChecksum16(recBuff, 15);
    if( (uint8)((checksumTotal / 256 ) & 0xff) != recBuff[5] )
    {
        printf("ConfigIO error : read buffer has bad checksum16(MSB)\n");
        return -1;
    }

    if( (uint8)(checksumTotal & 0xff) != recBuff[4] )
    {
        printf("ConfigIO error : read buffer has bad checksum16(LSB)\n");
```

```
        return -1;
    }

    if( extendedChecksum8(recBuff) != recBuff[0] )
    {
        printf("ConfigIO error : read buffer has bad checksum8\n");
        return -1;
    }

    if( recBuff[1] != (uint8)(0xF8) || recBuff[2] != (uint8)(0x05) || recBuff[3] != (uint8)(0x0B) )
    {
        printf("ConfigIO error : read buffer has wrong command bytes\n");
        return -1;
    }

    if( recBuff[6] != 0 )
    {
        printf("ConfigIO error : read buffer received errorcode %d\n", recBuff[6]);
        return -1;
    }

    if( recBuff[8] != 0 )
    {
        printf("ConfigIO error : NumberTimersEnabled was not set to 0\n");
        return -1;
    }

    if( recBuff[9] != 0 )
    {
        printf("ConfigIO error : CounterEnable was not set to 0\n");
        return -1;
    }

    return 0;
}

//Sends a StreamConfig low-level command to configure the stream.
int StreamConfig_example(HANDLE hDevice)
{
    int sendBuffSize;
    sendBuffSize = 14+NumChannels*2;
    uint8 sendBuff[sendBuffSize], recBuff[8];
    int sendChars, recChars;
    uint16 checksumTotal;
    uint16 scanInterval;
    int i;

    sendBuff[1] = (uint8)(0xF8);    //Command byte
```

```
sendBuff[2] = 4 + NumChannels;   //Number of data words = NumChannels + 4
sendBuff[3] = (uint8)(0x11);     //Extended command number
sendBuff[6] = NumChannels;       //NumChannels
sendBuff[7] = 1;                 //ResolutionIndex
sendBuff[8] = SamplesPerPacket;  //SamplesPerPacket
sendBuff[9] = 0;                 //Reserved
sendBuff[10] = 0;                //SettlingFactor: 0
sendBuff[11] = 0;  //ScanConfig:
//  Bit 3: Internal stream clock frequency = b0: 4 MHz
//  Bit 1: Divide Clock by 256 = b0

scanInterval = 4000;
sendBuff[12] = (uint8)(scanInterval&(0x00FF));  //scan interval (low byte)
sendBuff[13] = (uint8)(scanInterval/256);       //scan interval (high byte)

for( i = 0; i < NumChannels; i++ )
{
    sendBuff[14 + i*2] = i;  //ChannelNumber (Positive) = i
    sendBuff[15 + i*2] = 0;  //ChannelOptions:
    //  Bit 7: Differential = 0
    //  Bit 5-4: GainIndex = 0 (+-10V)
}

extendedChecksum(sendBuff, sendBuffSize);

//Sending command to U6
sendChars = LJUSB_Write(hDevice, sendBuff, sendBuffSize);
if( sendChars < sendBuffSize )
{
    if( sendChars == 0 )
        printf("Error : write failed (StreamConfig).\n");
    else
        printf("Error : did not write all of the buffer (StreamConfig).\n");
    return -1;
}

for( i = 0; i < 8; i++ )
    recBuff[i] = 0;

//Reading response from U6
recChars = LJUSB_Read(hDevice, recBuff, 8);
if( recChars < 8 )
{
    if( recChars == 0 )
        printf("Error : read failed (StreamConfig).\n");
    else
        printf("Error : did not read all of the buffer, %d (StreamConfig).\n", recChars);
```

```
        for( i = 0; i < 8; i++)
            printf("%d ", recBuff[i]);

        return -1;
    }

    checksumTotal = extendedChecksum16(recBuff, 8);
    if( (uint8)((checksumTotal / 256) & 0xff) != recBuff[5])
    {
        printf("Error : read buffer has bad checksum16(MSB) (StreamConfig).\n");
        return -1;
    }

    if( (uint8)(checksumTotal & 0xff) != recBuff[4] )
    {
        printf("Error : read buffer has bad checksum16(LSB) (StreamConfig).\n");
        return -1;
    }

    if( extendedChecksum8(recBuff) != recBuff[0] )
    {
        printf("Error : read buffer has bad checksum8 (StreamConfig).\n");
        return -1;
    }

    if( recBuff[1] != (uint8)(0xF8) || recBuff[2] != (uint8)(0x01) || recBuff[3] != (uint8)(0x11)
|| recBuff[7] != (uint8)(0x00) )
    {
        printf("Error : read buffer has wrong command bytes (StreamConfig).\n");
        return -1;
    }

    if( recBuff[6] != 0 )
    {
        printf("Errorcode # %d from StreamConfig read.\n", (unsigned int)recBuff[6]);
        return -1;
    }

    return 0;
}

//Sends a StreamStart low-level command to start streaming.
int StreamStart(HANDLE hDevice)
{
    uint8 sendBuff[2], recBuff[4];
    int sendChars, recChars;

    sendBuff[0] = (uint8)(0xA8);  //Checksum8
```

```
    sendBuff[1] = (uint8)(0xA8);  //Command byte

    //Sending command to U6
    sendChars = LJUSB_Write(hDevice, sendBuff, 2);
    if( sendChars < 2 )
    {
        if( sendChars == 0 )
            printf("Error : write failed.\n");
        else
            printf("Error : did not write all of the buffer.\n");
        return -1;
    }

    //Reading response from U6
    recChars = LJUSB_Read(hDevice, recBuff, 4);
    if( recChars < 4 )
    {
        if( recChars == 0 )
            printf("Error : read failed.\n");
        else
            printf("Error : did not read all of the buffer.\n");
        return -1;
    }

    if( normalChecksum8(recBuff, 4) != recBuff[0] )
    {
        printf("Error : read buffer has bad checksum8 (StreamStart).\n");
        return -1;
    }

    if( recBuff[1] != (uint8)(0xA9) || recBuff[3] != (uint8)(0x00) )
    {
        printf("Error : read buffer has wrong command bytes \n");
        return -1;
    }

    if( recBuff[2] != 0 )
    {
        printf("Errorcode # %d from StreamStart read.\n", (unsigned int)recBuff[2]);
        return -1;
    }

    return 0;
}

//Reads the StreamData low-level function response in a loop.
//All voltages from the stream are stored in the voltages 2D array.
int StreamData_example(HANDLE hDevice, u6CalibrationInfo *caliInfo)
```

```
{
    int recBuffSize;
    recBuffSize = 14 + SamplesPerPacket*2;
    int recChars, backLog;
    int i, j, k, m, packetCounter, currChannel, scanNumber;
    int totalPackets;  //The total number of StreamData responses read
    uint16 voltageBytes, checksumTotal;
    long startTime, endTime;
    int autoRecoveryOn;

    int numDisplay;         //Number of times to display streaming information
    int numReadsPerDisplay;  //Number of packets to read before displaying streaming
information
    int readSizeMultiplier;  //Multiplier for the StreamData receive buffer size
    int responseSize;        //The number of bytes in a StreamData response (differs with
SamplesPerPacket)

    numDisplay = 6;
    numReadsPerDisplay = 24;
    readSizeMultiplier = 5;
    responseSize = 14 + SamplesPerPacket*2;

    /* Each StreamData response contains (SamplesPerPacket / NumChannels) *
readSizeMultiplier
     * samples for each channel.
     * Total number of scans = (SamplesPerPacket / NumChannels) * readSizeMultiplier *
numReadsPerDisplay * numDisplay
     */
    double
voltages[(SamplesPerPacket/NumChannels)*readSizeMultiplier*numReadsPerDisplay*n
umDisplay][NumChannels];
    uint8 recBuff[responseSize*readSizeMultiplier];
    packetCounter = 0;
    currChannel = 0;
    scanNumber = 0;
    totalPackets = 0;
    recChars = 0;
    autoRecoveryOn = 0;

    printf("Reading Samples...\n");

    startTime = getTickCount();

    for( i = 0; i < numDisplay; i++ )
    {
        for( j = 0; j < numReadsPerDisplay; j++ )
        {
            /* For USB StreamData, use Endpoint 3 for reads.  You can read the multiple
```

```
         * StreamData responses of 64 bytes only if SamplesPerPacket is 25 to help
         * improve streaming performance.  In this example this multiple is adjusted
         * by the readSizeMultiplier variable.
         */

        //Reading stream response from U6
        recChars = LJUSB_Stream(hDevice, recBuff, responseSize*readSizeMultiplier);
        if( recChars < responseSize*readSizeMultiplier )
        {
            if(recChars == 0)
                printf("Error : read failed (StreamData).\n");
            else
                printf("Error : did not read all of the buffer, expected %d bytes but received
%d(StreamData).\n", responseSize*readSizeMultiplier, recChars);

            return -1;
        }

        //Checking for errors and getting data out of each StreamData response
        for( m = 0; m < readSizeMultiplier; m++ )
        {
            totalPackets++;

            checksumTotal = extendedChecksum16(recBuff + m*recBuffSize,
recBuffSize);
            if( (uint8)((checksumTotal >> 8) & 0xff) != recBuff[m*recBuffSize + 5] )
            {
                printf("Error : read buffer has bad checksum16(MSB) (StreamData).\n");
                return -1;
            }

            if( (uint8)(checksumTotal & 0xff) != recBuff[m*recBuffSize + 4] )
            {
                printf("Error : read buffer has bad checksum16(LSB) (StreamData).\n");
                return -1;
            }

            checksumTotal = extendedChecksum8(recBuff + m*recBuffSize);
            if( checksumTotal != recBuff[m*recBuffSize] )
            {
                printf("Error : read buffer has bad checksum8 (StreamData).\n");
                return -1;
            }

            if( recBuff[m*recBuffSize + 1] != (uint8)(0xF9) || recBuff[m*recBuffSize + 2] != 4
+ SamplesPerPacket || recBuff[m*recBuffSize + 3] != (uint8)(0xC0) )
            {
                printf("Error : read buffer has wrong command bytes (StreamData).\n");
```

```
            return -1;
        }

        if( recBuff[m*recBuffSize + 11] == 59 )
        {
            if( !autoRecoveryOn )
            {
                printf("\nU6 data buffer overflow detected in packet %d.\nNow using auto-
recovery and reading buffered samples.\n", totalPackets);
                autoRecoveryOn = 1;
            }
        }
        else if( recBuff[m*recBuffSize + 11] == 60 )
        {
            printf("Auto-recovery report in packet %d: %d scans were dropped.\nAuto-
recovery is now off.\n", totalPackets, recBuff[m*recBuffSize + 6] + recBuff[m*recBuffSize
+ 7]*256);
            autoRecoveryOn = 0;
        }
        else if( recBuff[m*recBuffSize + 11] != 0 )
        {
            printf("Errorcode # %d from StreamData read.\n", (unsigned int)recBuff[11]);
            return -1;
        }

        if( packetCounter != (int)recBuff[m*recBuffSize + 10] )
        {
            printf("PacketCounter (%d) does not match with with current packet count
(%d)(StreamData).\n", recBuff[m*recBuffSize + 10], packetCounter);
            return -1;
        }

        backLog = (int)recBuff[m*48 + 12 + SamplesPerPacket*2];

        for( k = 12; k < (12 + SamplesPerPacket*2); k += 2 )
        {
            voltageBytes = (uint16)recBuff[m*recBuffSize + k] +
(uint16)recBuff[m*recBuffSize + k+1]*256;

            getAinVoltCalibrated(caliInfo, 1, 0, 0, voltageBytes,
&(voltages[scanNumber][currChannel]));

            currChannel++;
            if( currChannel >= NumChannels )
            {
                currChannel = 0;
                scanNumber++;
            }
```

```
            }

            if(packetCounter >= 255)
               packetCounter = 0;
            else
               packetCounter++;
         }
      }

      printf("\nNumber of scans: %d\n", scanNumber);
      printf("Total packets read: %d\n", totalPackets);
      printf("Current PacketCounter: %d\n", ((packetCounter == 0) ? 255 :
packetCounter-1));
      printf("Current BackLog: %d\n", backLog);

      for( k = 0; k < NumChannels; k++ )
         printf("  AI%d: %.4f V\n", k, voltages[scanNumber - 1][k]);
   }

   endTime = getTickCount();
   printf("\nRate of samples: %.0lf samples per second\n",
(scanNumber*NumChannels)/((endTime - startTime)/1000.0));
   printf("Rate of scans: %.0lf scans per second\n\n", scanNumber/((endTime -
startTime)/1000.0));
   //printf(voltages);

   return 0;
}

//Sends a StreamStop low-level command to stop streaming.
int StreamStop(HANDLE hDevice)
{
   uint8 sendBuff[2], recBuff[4];
   int sendChars, recChars;

   sendBuff[0] = (uint8)(0xB0);  //Checksum8
   sendBuff[1] = (uint8)(0xB0);  //Command byte

   //Sending command to U6
   sendChars = LJUSB_Write(hDevice, sendBuff, 2);
   if( sendChars < 2 )
   {
      if( sendChars == 0 )
         printf("Error : write failed (StreamStop).\n");
      else
         printf("Error : did not write all of the buffer (StreamStop).\n");
      return -1;
   }
```

```
    //Reading response from U6
    recChars = LJUSB_Read(hDevice, recBuff, 4);
    if( recChars < 4 )
    {
        if( recChars == 0 )
            printf("Error : read failed (StreamStop).\n");
        else
            printf("Error : did not read all of the buffer (StreamStop).\n");
        return -1;
    }

    if( normalChecksum8(recBuff, 4) != recBuff[0] )
    {
        printf("Error : read buffer has bad checksum8 (StreamStop).\n");
        return -1;
    }

    if( recBuff[1] != (uint8)(0xB1) || recBuff[3] != (uint8)(0x00) )
    {
        printf("Error : read buffer has wrong command bytes (StreamStop).\n");
        return -1;
    }

    if( recBuff[2] != 0 )
    {
        printf("Errorcode # %d from StreamStop read.\n", (unsigned int)recBuff[2]);
        return -1;
    }

    /*
     //Reading left over data in stream endpoint.  Only needs to be done with firmwares
     //less than 0.94.
     uint8 recBuffS[64];
     int recCharsS = 64;
     printf("Reading left over data from stream endpoint.\n");
     while( recCharsS > 0 )
     recCharsS = LJUSB_Stream(hDevice, recBuffS, 64);
     */

    return 0;
}

void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
    int cmp;
```

```
    if (nrhs == 1)
    {
        if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
mxGetN(prhs[0])*mxGetM(prhs[0])!=1 )
        {
            mexErrMsgTxt("Input must be a scalar.");
        }
        cmp = (int)(mxGetScalar(prhs[0])+0.5);

        if (cmp == 0)
        {
                HANDLE hDevice;
                u6CalibrationInfo caliInfo;

                //Opening first found U6 over USB
                if( (hDevice = openUSBConnection(-1)) == NULL )
                    return;

                //Getting calibration information from U6
                if( getCalibrationInfo(hDevice, &caliInfo) < 0 )
                    closeUSBConnection(hDevice);

                if( ConfigIO_example(hDevice) != 0 )
                    closeUSBConnection(hDevice);

                //Stopping any previous streams
                StreamStop(hDevice);

                if( StreamConfig_example(hDevice) != 0 )
                    closeUSBConnection(hDevice);

                if( StreamStart(hDevice) != 0 )
                    closeUSBConnection(hDevice);

                StreamData_example(hDevice, &caliInfo);
                StreamStop(hDevice);

            closeUSBConnection(hDevice);

            return;
        }
        if (cmp == 1)
        {

            //HANDLE hDevice;
            //u6CalibrationInfo caliInfo;
            //int localID, numReadings;
            //long error;
```

```
        //numReadings = 5;

        //Open first found U6 over USB
        localID = -1;
        hDevice = openUSBConnection(localID);
        printf("Attempting to open U6...");
        if (hDevice == NULL)
        {
            //return;
            printf("failed.\n");
            mexErrMsgTxt("U6 not found\n");
        }
        printf("succeeded!\n");

        //Get calibration information from U6
        error = getCalibrationInfo(hDevice, &caliInfo);
        if( error < 0 )
            closeUSBConnection(hDevice);

        return;
    }
    if (cmp == 2)
    {
        plhs[0] = mxCreateDoubleMatrix(1,6,mxREAL);
        double *DAQdata = mxGetPr(plhs[0]);
/*
        HANDLE hDevice;
        u6CalibrationInfo caliInfo;
        int localID;
        long error;

        //Open first found U6 over USB
        localID = -1;
        hDevice = openUSBConnection(localID);
        printf("Attempting to open U6...");
        if (hDevice == NULL)
        {
            //return;
            printf("failed.\n");
            mexErrMsgTxt("U6 not found\n");
        }
        printf("succeeded!\n");

        //Get calibration information from U6
        error = getCalibrationInfo(hDevice, &caliInfo);
        if( error < 0 )
            closeUSBConnection(hDevice);
```

```
*/
        long error0, error1, error2, error3, error4, error5;
        double dblVoltage0, dblVoltage1, dblVoltage2, dblVoltage3, dblVoltage4,
dblVoltage5;

        //Read AIN3 single-ended voltage. +/-10 volt range and default resolution (0).
        //printf("\nCalling eAIN to read AIN voltage\n");

        //startTime = getTickCount();

        //for( i = 0; i < numReadings; i++ )
        //{
        dblVoltage0 = 0.0;
        dblVoltage1 = 0.0;
        dblVoltage2 = 0.0;
        dblVoltage3 = 0.0;
        dblVoltage4 = 0.0;
        dblVoltage5 = 0.0;
        error0 = eAIN(hDevice, &caliInfo, 0, 15, &dblVoltage0, LJ_rgBIP10V, 0, 0, 0, 0,
0);
        error1 = eAIN(hDevice, &caliInfo, 1, 15, &dblVoltage1, LJ_rgBIP10V, 0, 0, 0, 0,
0);
        error2 = eAIN(hDevice, &caliInfo, 2, 15, &dblVoltage2, LJ_rgBIP10V, 0, 0, 0, 0,
0);
        error3 = eAIN(hDevice, &caliInfo, 3, 15, &dblVoltage3, LJ_rgBIP10V, 0, 0, 0, 0,
0);
        error4 = eAIN(hDevice, &caliInfo, 4, 15, &dblVoltage4, LJ_rgBIP10V, 0, 0, 0, 0,
0);
        error5 = eAIN(hDevice, &caliInfo, 5, 15, &dblVoltage5, LJ_rgBIP10V, 0, 0, 0, 0,
0);
        if( error0 || error1 || error2 || error3 || error4 || error5 != 0 )
           closeUSBConnection(hDevice);

        *(DAQdata+0) = dblVoltage0;
        *(DAQdata+1) = dblVoltage1;
        *(DAQdata+2) = dblVoltage2;
        *(DAQdata+3) = dblVoltage3;
        *(DAQdata+4) = dblVoltage4;
        *(DAQdata+5) = dblVoltage5;

        //printf("AIN0 value = %.3f    ", dblVoltage0);
        //printf("AIN1 value = %.3f    ", dblVoltage1);
        //printf("AIN2 value = %.3f    \n", dblVoltage2);
        //sleep(0.1);
        //}

        return;
```

```
        }
        if (cmp == 3)
        {
            closeUSBConnection(hDevice);
            return;
        }
        else
        {
            mexErrMsgTxt("Not a valid control input value.");
        }

    }

    else
    {
        mexErrMsgTxt("Exactly one control input is required.");
    }

}
```

A.2: Matlab script to compile C interface

```
% Use this script to create the mex file using DAQinterfacer.c to
% interface the u6.c functions with Matlab.
% You have to copy the .mexmaci64 file wherever it was created and paste
% it to this directory.
% DAQinterfacer(0) is just a test to open and close the DAQ.
% This script only needs to be called if the mex file is missing or if
% some changes have been made in DAQinterfacer.c, requiring recompiling.
% Note: DAQinterfacer.c and DAQinterfacer2.c differ in that they use
% different approaches to read data from the DAQ. DAQinterfacer.c is
based
% on u6Stream.c and u6EFunctions.c while DAQinterfacer2.c is based on
% u6allio.c
%
% Mahtab Alam. Updated: Apr 2017.

close all
clear all

cd /Users/psychtests/Documents/LabJackU6DAQ/;
mex -I/Users/psychtests/Desktop/exodriver-master/liblabjackusb/
DAQinterfacer.c u6.c liblabjackusb-2.5.3.dylib;
DAQinterfacer(0);
```

A.3: Matlab function to open DAQ

```
function U6Open
% Use this to open the DAQ USB connection. With the mex file in the same
% directory, this function can access the DAQinterfacer.c file.
```

```
%
% Mahtab Alam. Updated: Apr 2017.

DAQinterfacer(1);
```

A.4: Matlab function to read from DAQ

```
function [U6Data,labels] = U6Read
% Use this to read from the DAQ USB connection. With the mex file in the
% same directory, this function can access the DAQinterfacer.c file. The
x,
% y, and z axes for the accelerometer are specified.
%
% Mahtab Alam. Updated: Apr 2017.

U6Data = DAQinterfacer(2);

labels = {'x','y','z','x','y','z'};
```

A.5: Matlab function to close DAQ

```
function U6Close
% Use this to close the DAQ USB connection. With the mex file in the
same
% directory, this function can access the DAQinterfacer.c file.
%
% Mahtab Alam. Updated: Apr 2017.

DAQinterfacer(3);
```

A.6: Matlab script to record and save DAQ data

```
%{
This script was written up to collect and save data from the
accelerometers. It was made compatible with the batch function in order
to
be able to run in the background with parallel computing. It's based on
the
U6DataCollect.m function. For use with the batch function, the function
was
removed and just made into a script. A few other changes were
implemented
as well:
(1) The plotting capability was removed.
(2) The tic and toc were replaced with a Psychtoolbox function GetSecs.
(3) Data will be saved to an Excel spreadsheet.

IMPORTANT: It is absolutely necessary to set the  variable "t" below
before
running this script. It will set the amount of time that data will be
collected for. This must be pre-defined. Exiting the program before
```

```
completion may result in data not being saved.

Usage:
Set the value of t as desired. Type in the Command Window "acc_job =
batch('U6DataCollect_forBatch');". This will save an object of class
'CJSIndependentJob' from which the arrays including all the data can be
extracted. (See glove_acc_stim_plots.m for a code snippet to do so,
using
fetchOutputs().) Additionally, all the data is saved in a .xls file.

Mahtab Alam. Updated: Apr 2017.
%}

close all

IncludePsychtoolbox('2014.10.beta')

min = 27;
t = 60*min; % This should be set to however long you think the session
will take.
f = 100;
% n = f*t;
s = 3;
% time_acc = ones(1,n);
% data_acc = ones(s,n);
% raw_time_acc = ones(1,n);
i = 1;

U6Open;

% start_time = GetSecs;
% for i = 1:n
%     U6Data = U6Read;
%     raw_time_acc(i) = GetSecs;
%     time_acc(i) = raw_time_acc(i) - start_time;
%     data_acc(:,i) = U6Data(1:s);
%     pause(1/f);
% end

start_time = GetSecs;
curr_time = GetSecs;
while curr_time < start_time + t
   raw_time_acc(i) = GetSecs;
   time_acc(i) = raw_time_acc(i) - start_time;
   U6Data = U6Read;
   data_acc(:,i) = U6Data(1:s);
   i = i + 1;
   WaitSecs(1/f);
   curr_time = GetSecs;
end

path = '~/Documents/ImagePres_DataCollect/Data/';
fid = fopen([path,'AccData_',datestr(now,'mmddyy-HHMM'),'.xls'],'a');
% fprintf(fid,'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n', 'Time (s)', 'GetSecs
(s)', ...
%     'AIN0', 'AIN1', 'AIN2', 'AIN3', 'AIN4', 'AIN5');
% for i = 1:length(raw_time_acc)
%     fprintf(fid,'%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n', time_acc(i),
raw_time_acc(i), data_acc(s*i-(s-1):s*i));
```

```
% end
fprintf(fid,'%s\t%s\t%s\t%s\t%s\n', 'Time (s)', 'Raw Time', ...
    'X', 'Y', 'Z');
for i = 1:length(raw_time_acc)
    fprintf(fid,'%d\t%.3f\t%d\t%d\t%d\n', time_acc(i), raw_time_acc(i),
data_acc(s*i-(s-1):s*i));
end


U6Close;
```

A.7: C++ code to interface gloves with Matlab

```
/*

This code serves to interface the Fifth Dimension Technologies Data Glove Ultra with
Matlab. The glove SDK provides C functions. This script takes the relevant C functions
and makes them compatible in Matlab. This code does not work on its own, but rather
needs to be compiled in and called from Matlab. Once compiled (and the mex file is
generated), this file does not need to be called on directly, as Matlab functions were
written (see below).

To use, do the following in Matlab:
   1. Navigate to the directory containing the "gloveInterfacer2.cpp" file.
   2. Compile on Matlab using the command "mex -L/usr/lib gloveInterfacer2.cpp
libfglove.so". *THIS IS FOR LINUX*
                        "mex -I/usr/local/include gloverInterfacer2.cpp fglove.dylib". *THIS IS
FOR MAC*
   3. Ensure that the file "gloveInterfacer2.mexa64" (Linux) or
"gloveInterfacer2.mexmaci64" (Mac) has been created.
   4. Run any of the following available commands:
      Input             Output
      ------------------------------------------------------------------------------------------------
      gloveInterfacer2(0)       Accesses the USB ports to register the gloves.
      gloveInterfacer2(1)       Captures the scaled glove data at an instant from all
sensors in both gloves.
      gloveInterfacer2(2)       Disconnects the gloves.
      gloveInterfacer2(3)       Determines which glove is the right hand (denoted by 1)
and left hand (denoted by 0).
      gloveInterfacer2(4)       Captures the raw uncalibrated glove data at an instant
from all sensors in both gloves.
      gloveInterfacer2(5)       Outputs a number indicating the gesture formed at that
instant. (See manual.)
      gloveInterfacer2(6)       ???
      gloveInterfacer2(7)       Get packet rate.

In order to call on these commands more easily and for use in Matlab scripts, functions
were created for each one as well. These functions simply run the desired
```

```
gloveInterfacer2 command.
 Command              Function file
 -------------------------------------
 gloveInterfacer2(0)   gloveOpen.m
 gloveInterfacer2(1)   gloveRead.m
 gloveInterfacer2(2)   gloveClose.m
 gloveInterfacer2(3)   gloveID.m
 gloveInterfacer2(4)   gloveReadRaw.m
 gloveInterfacer2(5)   gloveGesture.m


Mahtab Alam. Updated: Apr 2017.
 */

#include "fglove.h"
#include "mex.h"
#include <stdio.h>
#include <string.h>
#include <unistd.h>  // for usleep-----*/


char    *szPortA   = NULL;
char    *szPortB   = NULL;
fdGlove *pGloveA   = NULL;
fdGlove *pGloveB   = NULL;


int iNumCallbackCalls = 0;
float array[15000][14];


// this function will be registered as our callback function in the glove driver
void call_back(void* param)
{

    iNumCallbackCalls++;
    float gloveA_scaled[18];
    fdGetSensorScaledAll(static_cast<fdGlove*>(param), gloveA_scaled);

    array[iNumCallbackCalls][0] = gloveA_scaled[FD_THUMBNEAR];
    array[iNumCallbackCalls][1] = gloveA_scaled[FD_THUMBFAR];
    array[iNumCallbackCalls][2] = gloveA_scaled[FD_THUMBINDEX];
    array[iNumCallbackCalls][3] = gloveA_scaled[FD_INDEXNEAR];
    array[iNumCallbackCalls][4] = gloveA_scaled[FD_INDEXFAR];
    array[iNumCallbackCalls][5] = gloveA_scaled[FD_INDEXMIDDLE];
    array[iNumCallbackCalls][6] = gloveA_scaled[FD_MIDDLENEAR];
    array[iNumCallbackCalls][7] = gloveA_scaled[FD_MIDDLEFAR];
    array[iNumCallbackCalls][8] = gloveA_scaled[FD_MIDDLERING];
```

```
    array[iNumCallbackCalls][9] = gloveA_scaled[FD_RINGNEAR];
    array[iNumCallbackCalls][10] = gloveA_scaled[FD_RINGFAR];
    array[iNumCallbackCalls][11] = gloveA_scaled[FD_RINGLITTLE];
    array[iNumCallbackCalls][12] = gloveA_scaled[FD_LITTLENEAR];
    array[iNumCallbackCalls][13] = gloveA_scaled[FD_LITTLEFAR];

}



void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    int cmp;
    char str[80];
    sprintf(str,"%d\n",(int)(mxGetScalar(prhs[0])+0.5));

    if (nrhs == 1)
    {
        if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
mxGetN(prhs[0])*mxGetM(prhs[0])!=1 )
        {
            mexErrMsgTxt("Input must be a scalar.");
        }
        cmp = (int)(mxGetScalar(prhs[0])+0.5);

        if (cmp == 0) //This is the "gloveOpen.m" function.
        {
            int iNumCallbackCalls = 0;
            szPortA = "DG14U_L"; //This is the first glove that is plugged in.
            szPortB = "DG14U_R"; //This is the second glove that is plugged in.
            pGloveA = fdOpen(szPortA); //Open first glove - could be right or left.
            pGloveB = fdOpen(szPortB); //Open second glove - could be right or left.
            //pGloveA = fdOpen("DG14U_L"); //Open first glove - could be right or left.
            //pGloveB = fdOpen("DG14U_R"); //Open second glove - could be right or left.
            printf("Attempting to open first glove on %s ... ", szPortA);
            if (pGloveA == NULL)
            {
                printf("failed.\n"); //Can't detect first glove.
                //return;
                mexErrMsgTxt("Glove not found.\n");
            }
            printf("succeeded!\n"); //Detected first glove.

            printf("Attempting to open second glove on %s ... ", szPortB);
            if (pGloveB == NULL)
            {
                printf("failed.\n"); //Can't detect second glove.
                //return;
```

```
            mexErrMsgTxt("Glove not found.\n");
        }
        printf("succeeded!\n"); //Detected second glove.

        printf("Number of callback calls: %i\n", iNumCallbackCalls);

        return;
    }

    else if (cmp == 1) //This is the "gloveRead.m" function.
    {

        plhs[0] = mxCreateDoubleMatrix(1,28,mxREAL); //Create the 1x28 output array
for glove data.
        double *gloveData = mxGetPr(plhs[0]); //Pointer to output array.

        float gloveA_scaled[18];
        fdGetSensorScaledAll(pGloveA,gloveA_scaled);
        *(gloveData+0) = (double)gloveA_scaled[FD_THUMBNEAR];
        *(gloveData+1) = (double)gloveA_scaled[FD_THUMBFAR];
        *(gloveData+2) = (double)gloveA_scaled[FD_THUMBINDEX];
        *(gloveData+3) = (double)gloveA_scaled[FD_INDEXNEAR];
        *(gloveData+4) = (double)gloveA_scaled[FD_INDEXFAR];
        *(gloveData+5) = (double)gloveA_scaled[FD_INDEXMIDDLE];
        *(gloveData+6) = (double)gloveA_scaled[FD_MIDDLENEAR];
        *(gloveData+7) = (double)gloveA_scaled[FD_MIDDLEFAR];
        *(gloveData+8) = (double)gloveA_scaled[FD_MIDDLERING];
        *(gloveData+9) = (double)gloveA_scaled[FD_RINGNEAR];
        *(gloveData+10) = (double)gloveA_scaled[FD_RINGFAR];
        *(gloveData+11) = (double)gloveA_scaled[FD_RINGLITTLE];
        *(gloveData+12) = (double)gloveA_scaled[FD_LITTLENEAR];
        *(gloveData+13) = (double)gloveA_scaled[FD_LITTLEFAR];

        float gloveB_scaled[18];
        fdGetSensorScaledAll(pGloveB,gloveB_scaled);
        *(gloveData+14) = (double)gloveB_scaled[FD_THUMBNEAR];
        *(gloveData+15) = (double)gloveB_scaled[FD_THUMBFAR];
        *(gloveData+16) = (double)gloveB_scaled[FD_THUMBINDEX];
        *(gloveData+17) = (double)gloveB_scaled[FD_INDEXNEAR];
        *(gloveData+18) = (double)gloveB_scaled[FD_INDEXFAR];
        *(gloveData+19) = (double)gloveB_scaled[FD_INDEXMIDDLE];
        *(gloveData+20) = (double)gloveB_scaled[FD_MIDDLENEAR];
        *(gloveData+21) = (double)gloveB_scaled[FD_MIDDLEFAR];
        *(gloveData+22) = (double)gloveB_scaled[FD_MIDDLERING];
        *(gloveData+23) = (double)gloveB_scaled[FD_RINGNEAR];
        *(gloveData+24) = (double)gloveB_scaled[FD_RINGFAR];
        *(gloveData+25) = (double)gloveB_scaled[FD_RINGLITTLE];
        *(gloveData+26) = (double)gloveB_scaled[FD_LITTLENEAR];
```

```
        *(gloveData+27) = (double)gloveB_scaled[FD_LITTLEFAR];

    }

    else if (cmp == 2) //This is the "gloveClose.m" function.
    {
        //float gloveB_scaled[18];
        printf("close begin\n");
        printf("%d\t%d\n",pGloveA,pGloveB);
        fdClose(pGloveA); //Close first glove.
        //fdGetSensorScaledAll(pGloveB,gloveB_scaled);
        //printf("%f",(double)gloveB_scaled[FD_THUMBINDEX]);
        fdClose(pGloveB); //Close second glove.
        printf("close end\n");
    }

    else if (cmp == 3) //This is the "gloveID.m" function.
    {
        //          if (fdGetGloveHand(pGloveA) == 1)
        //          {
        //              printf("Glove A is right hand.\n");
        //          }
        //          else if (fdGetGloveHand(pGloveA) == 0)
        //          {
        //              printf("Glove A is left hand.\n");
        //          }
        //          if (fdGetGloveHand(pGloveB) == 1)
        //          {
        //              printf("Glove B is right hand.\n");
        //          }
        //          else if (fdGetGloveHand(pGloveB) == 0)
        //          {
        //              printf("Glove B is left hand.\n");
        //          }
        plhs[0] = mxCreateDoubleMatrix(1,2,mxREAL); //Create 1x2 output array for
handedness ID.
        double *gloveHand = mxGetPr(plhs[0]); //Pointer to output array.
        *(gloveHand+0) = (double)fdGetGloveHand(pGloveA);
        *(gloveHand+1) = (double)fdGetGloveHand(pGloveB);
    }

    else if (cmp == 4) //This is the "gloveReadRaw.m" function.
    {
        plhs[0] = mxCreateDoubleMatrix(1,28,mxREAL); //Create the 1x28 output array
for glove data.
        double *gloveData_raw = mxGetPr(plhs[0]); //Pointer to output array.

        unsigned short gloveA_raw[18];
```

```
    fdGetSensorRawAll(pGloveA,gloveA_raw);
    *(gloveData_raw+0) = (double)gloveA_raw[FD_THUMBNEAR];
    *(gloveData_raw+1) = (double)gloveA_raw[FD_THUMBFAR];
    *(gloveData_raw+2) = (double)gloveA_raw[FD_THUMBINDEX];
    *(gloveData_raw+3) = (double)gloveA_raw[FD_INDEXNEAR];
    *(gloveData_raw+4) = (double)gloveA_raw[FD_INDEXFAR];
    *(gloveData_raw+5) = (double)gloveA_raw[FD_INDEXMIDDLE];
    *(gloveData_raw+6) = (double)gloveA_raw[FD_MIDDLENEAR];
    *(gloveData_raw+7) = (double)gloveA_raw[FD_MIDDLEFAR];
    *(gloveData_raw+8) = (double)gloveA_raw[FD_MIDDLERING];
    *(gloveData_raw+9) = (double)gloveA_raw[FD_RINGNEAR];
    *(gloveData_raw+10) = (double)gloveA_raw[FD_RINGFAR];
    *(gloveData_raw+11) = (double)gloveA_raw[FD_RINGLITTLE];
    *(gloveData_raw+12) = (double)gloveA_raw[FD_LITTLENEAR];
    *(gloveData_raw+13) = (double)gloveA_raw[FD_LITTLEFAR];

    unsigned short gloveB_raw[18];
    fdGetSensorRawAll(pGloveB,gloveB_raw);
    *(gloveData_raw+14) = (double)gloveB_raw[FD_THUMBNEAR];
    *(gloveData_raw+15) = (double)gloveB_raw[FD_THUMBFAR];
    *(gloveData_raw+16) = (double)gloveB_raw[FD_THUMBINDEX];
    *(gloveData_raw+17) = (double)gloveB_raw[FD_INDEXNEAR];
    *(gloveData_raw+18) = (double)gloveB_raw[FD_INDEXFAR];
    *(gloveData_raw+19) = (double)gloveB_raw[FD_INDEXMIDDLE];
    *(gloveData_raw+20) = (double)gloveB_raw[FD_MIDDLENEAR];
    *(gloveData_raw+21) = (double)gloveB_raw[FD_MIDDLEFAR];
    *(gloveData_raw+22) = (double)gloveB_raw[FD_MIDDLERING];
    *(gloveData_raw+23) = (double)gloveB_raw[FD_RINGNEAR];
    *(gloveData_raw+24) = (double)gloveB_raw[FD_RINGFAR];
    *(gloveData_raw+25) = (double)gloveB_raw[FD_RINGLITTLE];
    *(gloveData_raw+26) = (double)gloveB_raw[FD_LITTLENEAR];
    *(gloveData_raw+27) = (double)gloveB_raw[FD_LITTLEFAR];
}

else if (cmp == 5) //This is the "gloveGesture.m" function.
{
    plhs[0] = mxCreateDoubleMatrix(1,2,mxREAL);
    double *gesture = mxGetPr(plhs[0]);
    *(gesture+0) = (double)fdGetGesture(pGloveA);
    *(gesture+1) = (double)fdGetGesture(pGloveB);

}

else if (cmp == 6)
{

    // Register callback function call_back() in glove driver pGloveA, and pass
    //  pGloveA as a void pointer to the callback function when it is called
```

```
        fdSetCallback(pGloveA, reinterpret_cast<void*>(&(call_back)),
static_cast<void*>(pGloveA));

        usleep(14000000); // sleep 5 seconds

        plhs[0] = mxCreateDoubleMatrix(14,iNumCallbackCalls,mxREAL); //Create the
iNumCallbackCallsx14 output array for glove data.
        double *gloveData = mxGetPr(plhs[0]); //Pointer to output array.

        for( int j=0; j<iNumCallbackCalls; j=j+1 )
           for( int i=0; i<14; i=i+1 )
              *(gloveData+((14*j)+i)) = (double)array[j][i];
        /*printf("\n%0.1f %0.1f %0.1f %0.1f %0.1f %0.1f %0.1f %0.1f %0.1f %0.1f
%0.1f %0.1f %0.1f",
         array[j][0],
         array[j][1],
         array[j][2],
         array[j][3],
         array[j][4],
         array[j][5],
         array[j][6],
         array[j][7],
         array[j][8],
         array[j][9],
         array[j][10],
         array[j][11],
         array[j][12],
         array[j][13]);*/

        // print the number of calls made to the callback function
        printf("\nNumber of callback calls: %i\n", iNumCallbackCalls);

        // Close gloves
        printf("\nClosing glove(s)...\n" );
        fdClose(pGloveA);
        fdClose(pGloveB);
        printf("\nGlove(s) closed.\n");

        return;
    }

    else if (cmp == 7) //Packet rate
    {
        plhs[0] = mxCreateDoubleMatrix(1,2,mxREAL);
        double *rate = mxGetPr(plhs[0]);
        *(rate+0) = (double)fdGetPacketRate(pGloveA);
        *(rate+1) = (double)fdGetPacketRate(pGloveB);
    }
```

```
    else
    {
       mexErrMsgTxt("Not a valid control input value.");
    }

  }

  else
  {
     mexErrMsgTxt("Exactly one control input is required.");
  }

}
```

A.8: Matlab script to compile C++ interface

```
% This script compiles the gloveInterfacer2.cpp file in Matlab. This
% must be done if any changes are made to that file. Afterwards it tests
% the glove is functioning properly by opening, recording data, then
% closing.
%
% Mahtab Alam. Updated: Apr 2017.

close
clear

cd /Users/malam/Documents/5DTDataGlove/
mex -I/usr/local/include gloveInterfacer2.cpp fglove.dylib

gloveOpen;
data = gloveRead;
gloveClose;
```

A.9: Matlab function to open data gloves

```
function gloveOpen
% This function initializes the glove. Data can not be acquired from the
% gloves before running this function. Do not use again if the gloves
are
% already initialized.
%
% Mahtab Alam. Updated: Apr 2017.

gloveInterfacer2(0);
```

A.10: Matlab function to identify glove handedness

```
function [gloveHand,labels] = gloveID
% This function returns 1 or 0 for each glove indicating whether it is
the
% left or right glove. A 0 means left glove and 1 means right glove. The
% outputs are listed in the order the gloves were plugged in. This can
be
% used to determine which glove is GloveA (plugged in first) and which
is
% GloveB (plugged in second).
%
% Mahtab Alam. Updated: Apr 2017.

gloveHand = gloveInterfacer2(3);

labels = {'GloveA','GloveB'};
```

A.11: Matlab function to identify glove gesture

```
function [gesture,labels] = gloveGesture
% This function returns a number indicating the hand gesture being
formed
% by each glove, based on the 5DT Data Glove manual.

gesture = gloveInterfacer2(5);

labels = {'GloveA','GloveB'};
```

A.12: Matlab function to read glove data

```
function [gloveData,sensorLabels] = gloveRead
% This function returns the glove data at a single time point. It can
not
% be called on without initializing the gloves first using gloveOpen.
The
% output is an array of 28 values, for the 14 sensors on each glove. The
% gloveA (plugged in first) values are listed first followed by gloveB
% (plugged in second). (You can use gloveID to determine which glove is
% which.) See the 5DT data glove manual for a diagram of the location of
% each sensor.
%
% Mahtab Alam. Updated: Apr 2017.

gloveData = gloveInterfacer2(1);

sensorLabels = {'thumb near','thumb far','thumb/index','index near',...
    'index far','index/middle','middle near','middle
far','middle/ring',...
    'ring near','ring far','ring/little','little near','little far'...
    'thumb near','thumb far','thumb/index','index near',...
    'index far','index/middle','middle near','middle
far','middle/ring',...
```

```
    'ring near','ring far','ring/little','little near','little far'};
```

A.13: Matlab function to close data gloves

```
function gloveClose
% This function disconnects the gloves from the computer. Do not use
this
% function if the gloves are not yet initialized with gloveOpen.
%
% Mahtab Alam. Updated: Apr 2017.

gloveInterfacer2(2);
```

A.14: Matlab script to plot glove data in real-time

```
% This plots the output from a single sensor in the glove in real-time.
The
% sensor of choice must be defined below by "s".
%
% Mahtab Alam. Updated: Apr 2017.

clear
close

nt = 100; % number of traces
np = 100; % number of readings per trace
s = 4; % sensor number

%%%%%%%%%%%%%%%%%%%%%%%%%% GLOVE %%%%%%%%%%%%%%%%%%%%%%%%%%

fig1 = figure(1);
axes('xlim',[1,np],'ylim',[0,1],...
    'gridlinestyle',':',...
    'ygrid','on',...
    'yminorgrid','on',...
    'minorgridlinestyle','-',...
    'ytick',0:s);
xlabel('Time (ms)');
ylabel('Sensor Number [Min: Open/Unflexed, Max: Closed/Flexed]');
title('Glove');
pos = get(fig1,'position');
set(fig1,'position',[pos(1:2)/4 pos(3)*1.5 pos(4)*2]);

x = 1:np; % sets the x-axis
lb1 = line([inf,inf],[0,1],'color','k'); % initializes vertical line

y=inf*ones(size(x));
lh=line(x,y,'marker','.','markersize',5,'linestyle','-');

%%%%%%%%%%%%%%%%%% Gather data and plot in real time %%%%%%%%%%%%%%%%%%

gloveOpen; % external function
glovehand = gloveID;
```

```
for i = 1:nt*np
    ix = rem(i-1,np)+1;
    glovedata = gloveRead; % external function outputs real time sensor
data
    glovedata1 = glovedata(s);
    y(ix) = glovedata1;
    set(lh,'ydata',y);
    if ix > 1
        if y(ix) - y(ix-1) > 0.05
            %set(lh,'Color','red');
            disp(y(ix));
            %elseif y(ix) <= 0.5
            %set(lh,'Color','blue');
        end
    end
    set(lb1,'xdata',[ix,ix]); % updates vertical line in real-time
    pause(.01); % a time-consuming operation
end

gloveClose; % external function
```

A.15: Matlab script to record and save glove data

```
%{
This script was written up to collect and save data from the gloves. It
was made compatible with the batch function in order to be able to run
in
the background with parallel computing. It's based on the DataCollect.m
function.
IMPORTANT: It is absolutely necessary to set the  variable "t" below
before
running this script. It will set the amount of time that data will be
collected for. This must be pre-defined. Exiting the program before
completion may result in data not being saved.

Usage:
Set the value of t as desired. Type in the Command Window "glove_job =
batch('DataCollect_forBatch');". This will save an object of class
'CJSIndependentJob' from which the arrays including all the data can be
extracted. (See glove_acc_stim_plots.m for a code snippet to do so,
using
fetchOutputs().) Additionally, all the data is saved in a .xls file.

Mahtab Alam. Updated: Apr 2017.
%}

close all

IncludePsychtoolbox('2014.10.beta') % Needed for GetSecs function

min = 27; % Set amount of time that data will be recorded for, in
minutes.
t = 60*min; % Set amount of time that data will be recorded for, in
```

```
seconds.
f = 130; % This is the "sampling rate" - note that it is not exact.
% n = f*t; % number of samples
s = 14; % Number of sensors per glove
i = 1;
% raw_time = ones(1,n);
% time = ones(1,n); % creates array for the timestamps data
% data_right = ones(s,n); % creates array for right glove data
% data_left = ones(s,n); % creates array for left glove data

gloveOpen; % external function to open gloves

glovehand = gloveID; % external function with handedness info


% Collect data from gloves and save them in matrices

start_time = GetSecs;
curr_time = GetSecs;
while curr_time < start_time + t
   raw_time(i) = GetSecs;
   time(i) = raw_time(i) - start_time;
   glovedata = gloveRead;
   glovegesture = gloveGesture;
   if glovehand(1) == 1 && glovehand(2) == 0 % determine L/R
        data_right(:,i) = glovedata(1:s);
        data_left(:,i) = glovedata(s+1:2*s);
        gesture_right(i) = glovegesture(1);
        gesture_left(i) = glovegesture(2);
    elseif glovehand(1) == 0 && glovehand(2) == 1 % determine L/R
        data_right(:,i) = glovedata(s+1:2*s);
        data_left(:,i) = glovedata(1:s);
        gesture_right(i) = glovegesture(2);
        gesture_left(i) = glovegesture(1);
   end
   i = i + 1;
   WaitSecs(1/f);
   curr_time = GetSecs;
end


% Remove repeated values from the recorded dataset and save.

time_left = time;
time_right = time;
raw_time_left = raw_time;
raw_time_right = raw_time;

for j = 2:length(time)
    if data_left(:,j-1) == data_left(:,j)
        data_left(:,j) = -1;
        time_left(j) = -1;
        raw_time_left(j) = -1;
    end
    if data_right(:,j-1) == data_right(:,j)
        data_right(:,j) = -1;
        time_right(j) = -1;
        raw_time_right(j) = -1;
    end
```

```
end

data_left = reshape(data_left(data_left~=-1),14,[]);
data_right = reshape(data_right(data_right~=-1),14,[]);
time_left = time_left(time_left~=-1);
time_right = time_right(time_right~=-1);
raw_time_left = raw_time_left(raw_time_left~=-1);
raw_time_right = raw_time_right(raw_time_right~=-1);


% Save the data to an Excel file.

path = '~/Documents/ImagePres_DataCollect/Data/';
fid = fopen([path,'GloveData_left_',datestr(now,'mmddyy-
HHMM'),'.xls'],'a');
fprintf(fid,
'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n',...
    'Times (s)', 'Raw Time', 'Sensor 0', 'Sensor 1', 'Sensor 2', 'Sensor
3',...
    'Sensor 4', 'Sensor 5', 'Sensor 6', 'Sensor 7', 'Sensor 8', 'Sensor
9',...
    'Sensor 10', 'Sensor 11', 'Sensor 12', 'Sensor 13');
for i = 1:length(raw_time_left)

fprintf(fid,'%d\t%.3f\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%
d\t%d\n', time_left(i), raw_time_left(i), data_left(s*i-(s-1):s*i));
end
fid = fopen([path,'GloveData_right_',datestr(now,'mmddyy-
HHMM'),'.xls'],'a');
fprintf(fid,
'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n',...
    'Times (s)', 'Raw Time', 'Sensor 0', 'Sensor 1', 'Sensor 2', 'Sensor
3',...
    'Sensor 4', 'Sensor 5', 'Sensor 6', 'Sensor 7', 'Sensor 8', 'Sensor
9',...
    'Sensor 10', 'Sensor 11', 'Sensor 12', 'Sensor 13');
for i = 1:length(raw_time_right)

fprintf(fid,'%d\t%.3f\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%
d\t%d\n', time_right(i), raw_time_right(i), data_right(s*i-(s-1):s*i));
end

gloveClose; % external function to close glove
```

A.16: Matlab function for behavioral test – class constructor

```
function t = ImagePresentation(p)
% ImagePresentation class constructor.
% t = ImagesPresentation creates ImagePresentation object using
parameters.
% Defines all object variables needed in the test. Must be defined here.
%
% Mahtab Alam. Updated: Apr 2017.

if nargin == 0
   t = class(NewTest(), mfilename);
elseif isa(p, mfilename)
```

```
   t = p;
else
   error(['Cannot create ' mfilename ' object from ' class(p) '
parameter']);
end


    function t = NewTest()

        % CONSTANT FIELDS

        t.base = '';

        % VARIABLE FIELDS
t.repeat = [];
        t.param = [];

        t.stim    = [];
        t.apert   = [];
        t.fx      = [];
        t.keyFlag = [];

        % Information for pregenerated stimuli.
        t.image = [];
        t.scrambled = [];

        t.numStimByCat = [];

        t.currTrial = [];

        global trialNum;
        trialNum = 0;


    end

end
```

A.17: Matlab function for behavioral test – parameter settings

```
function param = DefaultParameters(t)
%SETS ALL USER-DEFINED PARAMETERS USED TO DEFINE THE STIMULUS


%------TEST INFORMATION-------
testInfo.StructInfo.Title = 'Test Information';
testInfo.StructInfo.ElementNames = [];
testInfo.StructInfo.Help = [];
testInfo.StructInfo.EditFlag = 0;
testInfo.StructInfo.PrintFlag = [];
testInfo.StructInfo.MenuFlag = 0;
testInfo.Name = 'Image Presentation';
testInfo.FieldInfo.Name.Title = 'Test';
testInfo.Version = '201610';
```

```
testInfo.ShortName = 'ImgPres';


%-----STIMULUS PROPERTIES------
stimParam.StructInfo.Title = 'Stimulus Properties';
stimParam.StructInfo.ElementNames = [];
stimParam.StructInfo.Help = [];
stimParam.StructInfo.EditFlag = [];
stimParam.StructInfo.PrintFlag = [];
stimParam.StructInfo.MenuFlag = [];

stimParam.DelayNoFix = 2.5;
stimParam.FieldInfo.DelayNoFix.Title = 'Initial Delay before stimulus
without fixation (sec)';

%%%%%%%%%%%%%%THIS%%%%%%%%%%%%%%
% Delay during fixation
stimParam.Delay = 0.5;
stimParam.FieldInfo.Delay.Title = 'Second Delay before stimulus with
fixation (sec)';

stimParam.Duration = 5;
stimParam.FieldInfo.Duration.Title = 'Stimulus duration (sec)';

stimParam.TicksPerFrame = 1;
stimParam.FieldInfo.TicksPerFrame.Title = 'Video Ticks (video refreshes
per movie frame)';

stimParam.BackgndLum.RGBA = {42};
stimParam.BackgndLum.StructInfo.Title = 'Background Luminance (cd/m^2)';
                                        %luminance (Cd/m^2) cell
array; A single value specifies grayscale luminance
                                        %
A [r g b] triple specifies luminances for Red, Green and Blue separately

stimParam.AllowBreak = false;
stimParam.FieldInfo.AllowBreak.Title = 'Exit stimulus on key press';

stimParam.MaxRespTime = 100.0;
stimParam.FieldInfo.MaxRespTime.Title = 'Max. Response Time (sec)';

stimParam.FieldInfo.MaxRespTime.Help = [ ...
                                    'Maximum time
allowed to press a key'];

stimParam.NumApert = 1;
stimParam.FieldInfo.NumApert.MenuFlag = 0;   %DO NOT DISPLAY IN MENUS

stimParam.FieldInfo.NumApert.PrintFlag = 0;  %DO NOT PRINT IN OUTPUT

stimParam.optionPause='No';
stimParam.FieldInfo.optionPause.Title = 'Option for pause';

stimParam.FieldInfo.optionPause.Range.Values = {'Yes','No'};

stimParam.FieldInfo.optionPause.MenuFlag = 1;

daqs = DaqDeviceIndex;
```

```
if length(daqs) >= 1
    stimParam.Internal.DAQ = daqs(1);
    stimParam.Internal.DAQPort = 0;
    DaqDConfigPort(stimParam.Internal.DAQ, stimParam.Internal.DAQPort,
0);
else
    stimParam.Internal.DAQ = [];
end

stimParam.Internal.DAQStimulusOnsetMarker = 1;
%stimParam.Internal.DAQStimulusOffsetMarker = 2;


%-----IMAGE PROPERTIES--------
imgParam.StructInfo.Title = 'Image Properties';
imgParam.StructInfo.ElementNames = [];
imgParam.StructInfo.Help = [];
imgParam.StructInfo.EditFlag = [];
imgParam.StructInfo.PrintFlag = [];
imgParam.StructInfo.MenuFlag = [];

imgParam.ScaleToApert = false;
imgParam.FieldInfo.ScaleToApert.Title = 'Scale to aperture';

imgParam.FieldInfo.ScaleToApert.Help = [ ...
                                        'Whether to scale
all images to fill the aperture.  ' ...
                                        'If not, each image
will be its intrinsic pixel dimensions' ];

imgParam.RotateAng = 0;
imgParam.FieldInfo.RotateAng.Title = 'Rotate Angle (deg)';

imgParam.TransparentColor = '-1 -1 -1';
imgParam.FieldInfo.TransparentColor.Title = 'Tranparent Color (RGB)';

imgParam.FieldInfo.TransparentColor.Help = [ ...
                                        'Single color to
make transparent in all images. ' ...
                                        'Leave empty for
none.  (This operates in addition to any ' ...
                                        'alpha channel an
image has.)' ];

imgParam.ImageDir = ...
    fullfile(fileparts(mfilename('fullpath')),'objects');

imgParam.FieldInfo.ImageDir.Title = 'Image Path';

imgParam.FieldInfo.ImageDir.Button.Title = 'Browse';

imgParam.FieldInfo.ImageDir.Button.Callback = ...
                                        { 'GetDir', 'Select
image directory' };

MaxCats = 4;
imgParam.NumCat = 4;
imgParam.FieldInfo.NumCat.Title = 'Number of Categories';
```

```
imgParam.FieldInfo.NumCat.Range.Values = num2cell(1:MaxCats);

imgParam.FieldInfo.NumCat.Help = [ ...
                                                'Number of
categories represented by images ' ...
                                                '(up to '
num2str(MaxCats) ')' ];

imgParam.FieldInfo.NumCat.EditFlag = 2;

imgParam.NumCatAsPrimes = 3;
imgParam.FieldInfo.NumCatAsPrimes.Title = 'Number of Prime Categories';

imgParam.FieldInfo.NumCatAsPrimes.Range.Values = num2cell(1:MaxCats);

imgParam.FieldInfo.NumCatAsPrimes.Help = [ ...
                                                'Number of
categories that are primes ' ...
                                                '(up to '
num2str(MaxCats-1) ')' ];

imgParam.FieldInfo.NumCatAsPrimes.EditFlag = 2;

imgParam.CatName = {};
imgParam.FieldInfo.CatName.Title = 'Category Names';

imgParam.FieldInfo.CatName.EditFlag = 2;

imgParam.FieldInfo.CatName.MenuFlag = 0;

imgParam.CatFilePat = {'*.jpg'};
imgParam.FieldInfo.CatFilePat.Title = 'File Patterns';

imgParam.FieldInfo.CatFilePat.Help = [ ...
                                                'Filename patterns
specifying image files to use for ' ...
                                                'each category
(e.g., "*.jpg" or "square_*.gif")' ];

imgParam.BlockSize = 16;
imgParam.FieldInfo.BlockSize.Title = 'Block Size (pixels)';

imgParam.neutralTotal = 5;
imgParam.FieldInfo.neutralTotal.Title = 'No. of neutral images';

imgParam.FieldInfo.neutralTotal.Menuflag = 2;

imgParam.convertGray = false;
imgParam.FieldInfo.convertGray.Title = 'Use grayscale images';

imgParam.getnewMask = false;
imgParam.FieldInfo.getnewMask.Title = 'New Mask Required?';

imgParam.noofMasks = 0;
imgParam.FieldInfo.noofMasks.Title = 'No. of masks:';

imgParam.FieldInfo.noofMasks.MenuFlag = 0;

names = {'Power_Artefact', 'Power_Natural', 'Precision_Artefact',
```

```
'Precision_Natural'};
defaults = {'power_artefact_*', 'power_natural_*',
'precision_artefact_*', 'precision_natural_*'};

for j = 1:MaxCats
    if j<=length(names)
        imgParam.CatName{j} = names{j};
        imgParam.CatFilePat{j} = defaults{j};
    else
        imgParam.CatName{j} = ['Categ. ' num2str(j)];
        imgParam.CatFilePat{j} = '';
    end

end

imgParam.FieldInfo.CatName.ElementNames = imgParam.CatName;

imgParam.FieldInfo.CatFilePat.ElementNames = imgParam.CatName;


%-----APERTURE PROPERTIES--------
apertParam.StructInfo.Title = 'Aperture Properties';
apertParam.StructInfo.ElementNames = [];
apertParam.StructInfo.Help = [];
apertParam.StructInfo.EditFlag = [];
apertParam.StructInfo.PrintFlag = [];
apertParam.StructInfo.MenuFlag = [];

%NOTE: If only one value is given the aperture properties below are set
to
%the same value for all apertures. To specify unique values for each
aperture the
%parameter must be a 1xNumApert vector or cell array.

ApertNames = {'Image Apert.'};

apertParam.Shape = {'Rect'};
apertParam.FieldInfo.Shape.Title = 'Aperture Shape';

apertParam.FieldInfo.Shape.ElementNames = ApertNames;

apertParam.FieldInfo.Shape.MenuFlag = 0;       %DO NOT DISPLAY IN MENUS

apertParam.Eccen = [0];
apertParam.FieldInfo.Eccen.Title = 'Eccentricity (deg of visual angle)';

apertParam.FieldInfo.Eccen.ElementNames = ApertNames;

apertParam.Angle = [0];
apertParam.FieldInfo.Angle.Title = 'Polar angle (deg)';

apertParam.FieldInfo.Angle.ElementNames = ApertNames;

apertParam.Width = [12];
apertParam.FieldInfo.Width.Title = 'Width (deg of visual angle)';

apertParam.FieldInfo.Width.ElementNames = ApertNames;

apertParam.Height = [12];
```

```
apertParam.FieldInfo.Height.Title = 'Height (deg of visual angle)';

apertParam.FieldInfo.Height.ElementNames = ApertNames;


apertParam.MotionType = {'Planar'};
apertParam.FieldInfo.MotionType.Title = 'Type of Motion';

apertParam.FieldInfo.MotionType.ElementNames = ApertNames;

apertParam.FieldInfo.MotionType.Range.Values = {'None' 'Planar'};

apertParam.FieldInfo.MotionType.MenuFlag = 0;       %DO NOT DISPLAY IN
MENUS

apertParam.NoiseType = {'Random Walk'};
apertParam.FieldInfo.NoiseType.Title = 'Type of Noise';

apertParam.FieldInfo.NoiseType.ElementNames = ApertNames;

apertParam.FieldInfo.NoiseType.Range.Values = {'None' 'Flicker' 'Random
Walk'};

apertParam.FieldInfo.NoiseType.MenuFlag = 0;       %DO NOT DISPLAY IN
MENUS

apertParam.Coher = [NaN];
apertParam.FieldInfo.Coher.Title = 'Coherence';

apertParam.FieldInfo.Coher.ElementNames = ApertNames;

apertParam.FieldInfo.Coher.Range = [0 100];

apertParam.FieldInfo.Coher.MenuFlag = 0;       %DO NOT DISPLAY IN MENUS


apertParam.Lum.RGBA = ...
                  stimParam.BackgndLum.RGBA;
apertParam.Lum.StructInfo.Title = 'Aperture Luminance (Cd/m^2)';
                                                %luminance (Cd/m^2) cell
array; A single value specifies grayscale luminance
                                                %
A [r g b] triple specifies luminances for Red Green and Blue separately

apertParam.OuterDiamDeg = apertParam.Width;
apertParam.FieldInfo.OuterDiamDeg.Title = 'Outer diameter (deg of visual
angle)';

apertParam.FieldInfo.OuterDiamDeg.ElementNames = ApertNames;

apertParam.FieldInfo.OuterDiamDeg.MenuFlag = 0;       %DO NOT DISPLAY IN
MENUS

apertParam.InnerDiamDeg = [0];
apertParam.FieldInfo.InnerDiamDeg.Title = 'Inner diameter (deg of visual
angle)';

apertParam.FieldInfo.InnerDiamDeg.ElementNames = ApertNames;
```

```
apertParam.FieldInfo.InnerDiamDeg.MenuFlag = 0;       %DO NOT DISPLAY IN
MENUS


%-----FIXATION PROPERTIES-------
fxParam.StructInfo.Title = 'Fixation Properties';
fxParam.StructInfo.ElementNames = [];
fxParam.StructInfo.Help = [];
fxParam.StructInfo.EditFlag = [];
fxParam.StructInfo.PrintFlag = [];
fxParam.StructInfo.MenuFlag = [];

fxParam.Type = 'Cross';
fxParam.FieldInfo.Type.Title = 'Type of Fixation';

fxParam.Dim = [100 100];
fxParam.FieldInfo.Dim.Title = 'Dimensions (minutes)';

fxParam.FieldInfo.Dim.ElementNames = {'Horiz.', 'Vertical'};

fxParam.Lum.RGBA = {[80 0 0]};
fxParam.Lum.StructInfo.Title = 'Fixation Luminance (Cd/m^2)';
                                          %luminance (Cd/m^2) cell
array; A single value specifies grayscale luminance
                                          %
A [r g b] triple specifies luminances for Red Green and Blue separately

fxParam.Loc = [0 0];
fxParam.FieldInfo.Loc.Title = 'Location (deg)';

fxParam.FieldInfo.Loc.ElementNames = {'Eccen.', 'Polar Angle'};
                                          %1x2 vector specifying
the eccentricity (deg) and
                                          %polar angle (deg)
relative to the screen center.
                                          %Default is screen
center = [0 0].

fxParam.PenWidth = 4;
fxParam.FieldInfo.PenWidth.Title = 'Pen width (minutes)';

fxParam.Filled = false;
fxParam.FieldInfo.Filled.Title = 'Fill shape';


%-------RESPONSE KEYS-------------
keyInfo.StructInfo.Title = 'Key Definitions';
keyInfo.StructInfo.ElementNames = [];
keyInfo.StructInfo.Help = [];
keyInfo.StructInfo.EditFlag = [];
keyInfo.StructInfo.PrintFlag = [];

keyInfo.Quit = {'q', 'ESCAPE'};
keyInfo.FieldInfo.Quit.Title = 'Quit';

keyInfo.FieldInfo.Quit.EditFlag = 0;

keyInfo.Repeat = 'r';
```

```
keyInfo.FieldInfo.Repeat.Title = 'Repeat';

keyInfo.FieldInfo.Repeat.EditFlag = 0;

keyInfo.Help = 'h';
keyInfo.FieldInfo.Help.Title = 'Help';

keyInfo.FieldInfo.Help.EditFlag = 0;

keyInfo.Answer = {'1', '2',  ...
                  '1!', '2@'};
keyInfo.FieldInfo.Answer.Title = 'Test Answer';
                                                %strings with symbols
after numbers (e.g., '1!') denote
                                                %keyboard numeric key as
opposed to the keypad

keyInfo.AnswerMap = [1 2 1 2];
keyInfo.FieldInfo.AnswerMap.Title = 'Answer Category';

keyInfo.FieldInfo.AnswerMap.Range.Values = num2cell(1:4);

keyInfo.FieldInfo.AnswerMap.Help = [ ...
                                                'Categories
(represented as numbers) to which each ' ...
                                                'corresponding
answer key maps' ];

ansKeyNames = arrayfun(@(d) '', 1:length(keyInfo.Answer),
'UniformOutput', false);

keyInfo.FieldInfo.Answer.ElementNames = ansKeyNames;

keyInfo.FieldInfo.AnswerMap.ElementNames = ansKeyNames;


%-------PARADIGMS-------------
paradigm.StructInfo.Title = 'Experimental Paradigm';
paradigm.StructInfo.ElementNames = [];
paradigm.StructInfo.Help = [];
paradigm.StructInfo.EditFlag = [];
paradigm.StructInfo.PrintFlag = [];
paradigm.StructInfo.MenuFlag = [];

paradigm.Type = 'CS';
paradigm.FieldInfo.Type.Range.Values = { 'CS' };

paradigm.CS = CSParadigm();
csParam = GetParams(paradigm.CS);
csParam.Levels.PrimeType = 1:4; %Number of categories
csParam.Levels.stimulus = 1:26; %Number of objects per category
%csParam.Levels.stimulus = 1:4;
%csParam.Levels.TestType = [1:20];
%%%%%%%%%%%%%%THIS%%%%%%%%%%%%%%
% Duration of object image
csParam.Levels.PrimeDuration = 1500;
csParam.Levels.FieldInfo.PrimeDuration.Title   = 'Prime Duration
(msec)';
csParam.Levels.SOA = 0;
```

```
csParam.Levels.FieldInfo.SOA.Title   = 'SOA (msec)';
csParam.Levels.MaskDuration = 0;
csParam.Levels.FieldInfo.MaskDuration.Title = 'Mask Duration (msec)';
csParam.TrialsPerLevel = 1;
csParam.ApplyMaxSameTo = {'PrimeType'};

csParam.FieldInfo.ApplyMaxSameTo.MenuFlag = 1;

csParam.FieldInfo.ApplyMaxSameTo.PrintFlag = 0;
csParam.SummarizeBy = {'PrimeType' 'SOA'};

csParam.FieldInfo.SummarizeBy.MenuFlag = 1;

csParam.FieldInfo.SummarizeBy.PrintFlag = 0;
paradigm.CS = SetParams(paradigm.CS, csParam);


%-------PARAMETER RANGE AND HELP DESCRIPTIONS------------
SetCommonRangeValues;
fxParam.FieldInfo.Type.Range.Values = @(s,f) MarkDrawer('Types');
SetCommonHelpFields;


%--------PACK PARAMETERS INTO SINGLE STRUCTURE-----------
param.StructInfo.Title = [testInfo.Name ' Parameters'];
param.StructInfo.ElementNames = [];
param.StructInfo.Help = [];
param.StructInfo.EditFlag = 0;
param.StructInfo.PrintFlag = [];
param.StructInfo.MenuFlag = 0;

param.testInfo = testInfo;
param.paradigm = paradigm;
param.stimParam = stimParam;
param.stimParam.imgParam = imgParam;
param.stimParam.apertParam = apertParam;
param.fxParam = fxParam;
param.keyInfo = keyInfo;
```

A.18: Matlab function for behavioral test – display stimuli

```
function [t, presenter, key, trialOutput] = DisplayNextStim(t,
windowPtr, scrn, ...
                                                        endTime,
presenter, ...

keyDeviceIndices, key)
% Displays the appropriate stimulus at the designated time.
%
% Mahtab Alam. Updated: Apr 2017.

    global trialNum;
    trialNum = trialNum + 1;

    trialStartTime = GetSecs;
    [t.param.paradigm.(t.param.paradigm.Type), level] = ...
        GetCurrentLevel(t.param.paradigm.(t.param.paradigm.Type));
```

```
    if isempty(t.repeat)
        t.currTrial.level = level;
        % No neutral image repeated more than 5 times
        while true
            listNum =
randint(1,1,t.param.stimParam.imgParam.neutralTotal);% + 1;
            if
(sum(find(listNum==t.param.stimParam.imgParam.neutralList))==0)
                level.TestType = listNum;
                t.param.stimParam.imgParam.neutralList(end) = listNum;
                t.param.stimParam.imgParam.neutralList =
circshift(t.param.stimParam.imgParam.neutralList,[1 1]);
                break;
            end
        end
        t.currTrial.level.TestType = level.TestType;
    else
        t.currTrial = t.repeat;
        level = t.currTrial.level;
    end

    %WaitSecs(t.param.stimParam.Delay);
    %Show fixation point.
    [presenter] = DrawLoop(presenter, ...
                      windowPtr, scrn, t.stim, ...
                      {}, ...
                      trialStartTime + t.param.stimParam.Delay, ...
                      keyDeviceIndices);

%     %WaitSecs(t.param.stimParam.DelayNoFix);
%     % Delay before stimulus.
%     [presenter] = DrawLoopNoFixation(presenter, ...
%                       windowPtr, scrn, t.stim, ...
%                       {}, ...
%                       trialStartTime + t.param.stimParam.Delay+
t.param.stimParam.DelayNoFix, ...
%                       keyDeviceIndices);

    trialStartTime = GetSecs;

    if ~isempty(t.param.stimParam.Internal.DAQ)
        err =
DaqDOut(t.param.stimParam.Internal.DAQ,t.param.stimParam.Internal.DAQPor
t,t.param.stimParam.Internal.DAQStimulusOnsetMarker);
        err =
DaqDOut(t.param.stimParam.Internal.DAQ,t.param.stimParam.Internal.DAQPor
t,0);
    end


    if t.stim.AllowBreak
        keyMode = 'breakonkey';
    else
        keyMode = [];
    end

    % Show stimulus.
    shiftImgNum = 0;
```

```
    for i = 1:level.PrimeType-1
        shiftImgNum = shiftImgNum + t.numStimByCat(i);
    end

    primeImgNum = shiftImgNum + level.stimulus;

    fps = scrn.FrameRate/t.param.stimParam.TicksPerFrame;

    if t.param.stimParam.imgParam.ScaleToApert
         primeSize = round([t.apert.Width t.apert.Height]);
    else
        primeSize = [t.image(primeImgNum).Height
t.image(primeImgNum).Width];
    end

    image1(:,:,1:ceil((level.PrimeDuration/1000)*fps)) =
t.image(primeImgNum).Texture;

    t.currTrial.displayStartTime = GetSecs;
    prevStartTime = GetSecs;

    % Show Prime.
     [presenter, drawer, key] = DrawLoopNoFixation(presenter, ...
        windowPtr, scrn, t.stim, ...
        { ApertDrawer(t.apert, ...
                      ones(1,length(t.apert))) ...
          TexturesDrawer(image1, primeSize, ...
            t.apert.Center(1), t.apert.Center(2), [], ...
            t.param.stimParam.imgParam.RotateAng) ...
        }, ...
        t.currTrial.displayStartTime + level.PrimeDuration/1000, ...
        keyDeviceIndices, keyMode);
    primeDuration = GetSecs - prevStartTime;
    %prevStartTime = GetSecs;

    %Cue to form grip
    gripStartTime = GetSecs;
    [presenter] = DrawLoopNoFixation(presenter, ...
                      windowPtr, scrn, t.stim, ...
                      {StringDrawer('Form grip', scrn.Width/2,
scrn.Height/2, 40, ...
                                    [42 42 42])}, ...
                      gripStartTime + 1.5, ...
                      keyDeviceIndices);

    trialOutput = sprintf('%.3f\t%d\t%d\t%s\t%d', trialStartTime,
level.PrimeType, ...
        level.stimulus, t.image(primeImgNum).FileName,
round(primeDuration*1000));

    time = GetSecs;
    if mod(trialNum, 52) == 0
        keyMode = 'breakonkey';
        instruction_string = 'Rest for up to 1 min. Press 1 to continue
at any time.';
        [presenter] = DrawLoopNoFixation(presenter, windowPtr, scrn, ...
            t.stim, {StringDrawer(instruction_string, scrn.Width/2,
scrn.Height/2, 40, ...
                                    [42 42 42])}, time + 60,
```

```
keyDeviceIndices, keyMode);
    end

end
```

A.19: Matlab function for behavioral test – key response

```
function [t, presenter, key, trialOutput, status] = ...
    WaitForKeyResponse(t, windowPtr, scrn, ...
                       endTime, presenter, ...
                       keyDeviceIndices, key)
% Waits for keypress (or lack of one) until the end of display time for
% each stimulus. Processes user's key response and makes an evauation
% based on task-key mapping.
%
% Mahtab Alam. Updated: Apr 2017.

    status = '';

    displayStartTime = t.currTrial.displayStartTime;

    [t.param.paradigm.(t.param.paradigm.Type), level] = ...
        GetCurrentLevel(t.param.paradigm.(t.param.paradigm.Type));

    msg = '';

    if isempty(key)
        fprintf(1,'\n%g\t%g\n',t.param.stimParam.MaxRespTime,endTime-
GetSecs);
        [presenter, drawer, key] = DrawLoopNoFixation(presenter, ...
            windowPtr, scrn, t.stim, ...
            {}, ...
            min(endTime, GetSecs+t.param.stimParam.MaxRespTime), ...
            keyDeviceIndices, []);

    end
    if ~isempty(key) && nnz(key(1).Code) > 0
        keyStr = KbName(find(key(1).Code));
        reactionTime = key(1).PressTime - displayStartTime;
    else
        keyStr = '';
        reactionTime = NaN;
        key.Code = -1;
        endTime = 0;
    end
    t.repeat = [];
    % Map key to corresponding answer.
    ansKeyIndex = find(strcmp(keyStr, t.param.keyInfo.Answer), 1);  %
use first if multiple
    ansIndex = t.param.keyInfo.AnswerMap(ansKeyIndex);

    if nnz(key(1).Code) > 1  % not time limited - invalid key pressed
        msg = [msg 'Invalid key-trial repeated.'];
        status = 'repeat';
    elseif ismember(keyStr, t.param.keyInfo.Repeat)
```

```
        msg = [msg 'Repeat key-trial repeated.'];
        status = 'repeat';
    elseif ismember(keyStr, t.param.keyInfo.Help)
        msg = [msg 'Help key, repeating trial.'];
        status = 'help';
        t.currTrial.help = 1;
    elseif ismember(keyStr, t.param.keyInfo.Quit)
        msg = [msg 'Quit key-test quit.'];
        status = 'quit';
    elseif ~isempty(ansIndex) && ...
            ansIndex > 0 && ansIndex <=
t.param.stimParam.imgParam.NumCat
        % A valid answer key was pressed
        evaluation = ansIndex == ceil(level.PrimeType/2);

        if ansIndex == 1
            msg = [msg sprintf('%s\t%d\t%.3f', ...
                    'Power', ...
                    evaluation, reactionTime)];
        elseif ansIndex == 2
            msg = [msg sprintf('%s\t%d\t%.3f', ...
                    'Precision', ...
                    evaluation, reactionTime)];
        end

        t.param.paradigm.(t.param.paradigm.Type) = Record( ...
            t.param.paradigm.(t.param.paradigm.Type), ...
            all(evaluation == 1) ...  % all answers correct?
        );

    elseif isempty(endTime)  % not time limited - invalid key pressed
        msg = [msg 'Invalid key-trial repeated.'];
        status = 'repeat';
    elseif isempty(keyStr)   % time limited - no key pressed
        msg = [msg 'No response-timed out.'];
    else                     % time limited - invalid key pressed
        msg = [msg 'Invalid key "' keyStr '"'];
    end

    % Generate string of results.

    tabsInAnsOutput = 2;
    trialOutput = sprintf('\t%s%s', ...
        msg, ...
        repmat(sprintf('\t'), 1, tabsInAnsOutput-length(regexp(msg,
'\t'))) ... % pad tabs
    );
    %trialOutput = [trialOutput sprintf('\t%.3f', displayStartTime)];

    switch status
        case 'quit'
            return;
        case 'repeat'
            t.repeat = t.currTrial;
        case 'help'
            t.repeat = t.currTrial;
        otherwise
            % Advance paradigm to next trial
            [t.param.paradigm.(t.param.paradigm.Type)] = ...
                AdvanceLevel(t.param.paradigm.(t.param.paradigm.Type));
```

```
            if (strcmp(t.param.stimParam.optionPause,'Yes'))
                keyMode = 'breakonkey';
                pause(1);

               [presenter, drawer, key] = DrawLoopNoFixation(presenter,
...
                    windowPtr, scrn, t.stim, {   ApertDrawer(t.apert,
...
                                ones(1,length(t.apert))) ...
                                StringDrawer(' ! ', t.apert.Center(1),
t.apert.Center(2), 80, ...
                                    [42 42 42])}, min(endTime, GetSecs +
t.param.stimParam.Duration), ...
                keyDeviceIndices, keyMode);
            end

            if Done(t.param.paradigm.(t.param.paradigm.Type))
                status = 'done';
            end
    end

    if strcmp(status,'done')
        WaitSecs(1);

            [triggerTime, runAgainKey] = WaitForTrigger(windowPtr, {'1',
'2', '1!', '2@'}, sprintf('Run again?\n1=yes, 2=no'), 255, t.stim.Color,
0);
            if strcmp(KbName(find(runAgainKey.Code)), '1') ||
strcmp(KbName(find(runAgainKey.Code)), '1!')
                t = Reset2(t);
                status = '';
            else
                status = 'done';
            end

    end

    % Wait out rest of period.
%     [presenter] = DrawLoopNoFixation(presenter, windowPtr, scrn,
t.stim, ...
%         {}, endTime, keyDeviceIndices);
%    while GetSecs < min(endTime, GetSecs +
t.param.stimParam.MaxRespTime)
        WaitSecs(0.5+(1-0.5).*rand(1,1)); % set the amount of time to
wait between trials
%    end

end
```

A.20: Matlab script for running all components

```
% The commands to run all three components simultaneously. Make sure t
% values are set beforehand in both the glove and DAQ files.
```

```
%
% Mahtab Alam. Updated: April 2017.

clear
cd /Users/psychtests/Documents/ImagePres_DataCollect/

glove_job = batch('DataCollect_forBatch');
acc_job = batch('U6DataCollect_forBatch');

%BraviShell %run if BraviShell is not already initialized
BraviShell ImagePresentation
```

BIBLIOGRAPHY

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(4), 433–459.

Borghi, A. M., Bonfiglioli, C., Lugli, L., Ricciardelli, P., Rubichi, S., & Nicoletti, R. (2007). Are visual stimuli sufficient to evoke motor information?: Studies with hand primes. *Neuroscience Letters*, *411*(1), 17–21.

Bourguignon, M., De Tiège, X., de Beeck, M. O., Van Bogaert, P., Goldman, S., Jousmäki, V., & Hari, R. (2013). Primary motor cortex and cerebellum are coupled with the kinematics of observed hand movements. *NeuroImage*, *66*, 500–507.

Bourguignon, M., Jousmäki, V., Op de Beeck, M., Van Bogaert, P., Goldman, S., & De Tiège, X. (2012). Neuronal network coherent with hand kinematics during fast repetitive hand movements. *NeuroImage*, *59*(2), 1684–1691.

Carlson, L., & Van der Zee, E. (Eds.) (2004). *Functional features in language and space*. Oxford: Oxford University Press.

Chestek, C. A., Gilja, V., Blabe, C. H., Foster, B. L., Shenoy, K. V., Josef Parvizi, & Henderson, J. M. (2013). Hand posture classification using electrocorticography signals in the gamma band over human sensorimotor brain areas. *Journal of Neural Engineering*, *10*(2), 26002.

Cruz, E. G., Waldinger, H. C., & Kamper, D. G. (2005). Kinetic and kinematic workspaces of the index finger following stroke. *Brain*, *128*(5), 1112–1121.

Dale, A. M., Fischl, B., & Sereno, M. I. (1999). Cortical Surface-Based Analysis: I. Segmentation and Surface Reconstruction. *NeuroImage*, *9*(2), 179–194.

Dale, A. M., Liu, A. K., Fischl, B. R., Buckner, R. L., Belliveau, J. W., Lewine, J. D., & Halgren, E. (2000). Dynamic Statistical Parametric Mapping: Combining fMRI and MEG for High-Resolution Imaging of Cortical Activity. *Neuron*, 26(1), 55–67.

Destrieux, C., Fischl, B., Dale, A., & Halgren, E. (2010). Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. *NeuroImage,* 53(1), 1–15.

Ehrsson, H. H., Fagergren, A., Jonsson, T., Westling, G., Johansson, R. S., & Forssberg, H. (2000). Cortical Activity in Precision- Versus Power-Grip Tasks: An fMRI Study. *Journal of Neurophysiology*, *83*(1), 528–536.

Fischl, B., Salat, D. H., Busa, E., Albert, M., Dieterich, M., Haselgrove, C., … Dale, A. M. (2002). Whole Brain Segmentation: Automated Labeling of Neuroanatomical Structures in the Human Brain. *Neuron*, *33*(3), 341–355.

Fischl, B., Sereno, M. I., & Dale, A. M. (1999). Cortical Surface-Based Analysis: II: Inflation, Flattening, and a Surface-Based Coordinate System. *NeuroImage*, *9*(2), 195–207.

Fischl, B., van der Kouwe, A., Destrieux, C., Halgren, E., Ségonne, F., Salat, D. H., … Dale, A. M. (2004). Automatically Parcellating the Human Cerebral Cortex. *Cerebral Cortex*, *14*(1), 11–22.

Gonzalez, C. L. R., Ganel, T., & Goodale, M. A. (2006). Hemispheric Specialization for the Visual Control of Action Is Independent of Handedness. *Journal of Neurophysiology*, *95*(6), 3496–3501.

Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., … Hämäläinen, M. S. (2014). MNE software for processing MEG and EEG data. *NeuroImage*, *86*, 446–460.

Grèzes, J., Tucker, M., Armony, J., Ellis, R., & Passingham, R. E. (2003). Objects automatically potentiate action: an fMRI study of implicit processing. *European Journal of Neuroscience*, *17*(12), 2735–2740.

Hämäläinen, M., Hari, R., Ilmoniemi, R. J., Knuutila, J., & Lounasmaa, O. V. (1993). Magnetoencephalography - theory, instrumentation, and applications to noninvasive studies of the working human brain. *Reviews of Modern Physics*, *65*(2), 413–497.

Hanson, T. L., Fuller, A. M., Lebedev, M. A., Turner, D. A., & Nicolelis, M. A. L. (2012). Subcortical Neuronal Ensembles: An Analysis of Motor Task Association, Tremor, Oscillations, and Synchrony in Human Patients. *Journal of Neuroscience*, *32*(25), 8620–8632.

Huber, M., Rabin, B., Docan, C., Burdea, G. C., AbdelBaky, M., & Golomb, M. R. (2010). Feasibility of Modified Remotely Monitored In-Home Gaming Technology for Improving Hand Function in Adolescents With Cerebral Palsy. *IEEE Transactions on Information Technology in Biomedicine*, *14*(2), 526–534.

Kamel, N. S., & Sayeed, S. (2008). Svd-based signature verification technique using data glove. *International Journal of Pattern Recognition and Artificial Intelligence*, *22*(3), 431–443.

Kamper, D. g., & Rymer, W. z. (2001). Impairment of voluntary control of finger motion following stroke: Role of inappropriate muscle coactivation. *Muscle & Nerve*, *24*(5), 673–681.

Kandel, E.R., Schwartz, J.H., & Jessell, T.M. (Eds.) (2000). *Principles of neural science*. New York: McGraw-Hill.

Letham, B., & Raij, T. (2011). Statistically robust measurement of evoked response onset latencies. *Journal of Neuroscience Methods*, *194*(2), 374–379.

Linden, D. E. J., & Turner, D. L. (2016). Real-time functional magnetic resonance imaging neurofeedback in motor neurorehabilitation. *Current Opinion in Neurology*, *29*(4), 412–418.

Muir, R. B., & Lemon, R. N. (1983). Corticospinal neurons with a special role in precision grip. *Brain Research*, *261*(2), 312–316.

Napier, J. R. (1956). The Prehensile Movements of the Human Hand. *Bone & Joint Journal*, *38–B*(4), 902–913.

Nowak, D. A., Grefkes, C., Dafotakis, M., Küst, J., Karbe, H., & Fink, G. R. (2007). Dexterity is impaired at both hands following unilateral subcortical middle cerebral artery stroke. *European Journal of Neuroscience*, *25*(10), 3173–3184.

Nowak, D. A., & Hermsdörfer, J. (2003). Selective deficits of grip force control during object manipulation in patients with reduced sensibility of the grasping digits. *Neuroscience Research*, *47*(1), 65–72.

Pedley, M. (2013). AN 3461 Application Note: Tilt Sensing Using a Three-Axis Accelerometer. https://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf

Piitulainen, H., Bourguignon, M., De Tiège, X., Hari, R., & Jousmäki, V. (2013). Corticokinematic coherence during active and passive finger movements. *Neuroscience*, *238*, 361–370.

Rajendran, K., Samraj, A., & Rajavel, M. (2013). Emergency gesture communication by patients, elderly and differently abled with care takers using wearable data gloves. *Journal of Signal and Information Processing*, *4*(1), 1–9. doi: 10.4236/jsip.2013.41001

Roland, P. E., & Zilles, K. (1996). Functions and structures of the motor cortices in humans. *Current Opinion in Neurobiology*, *6*(6), 773–781.

Tadel, F., Baillet, S., Mosher, J. C., Pantazis, D., & Leahy, R. M. (2011). Brainstorm: A User-Friendly Application for MEG/EEG Analysis. *Computational Intelligence and Neuroscience*, *2011*, e879716.

Vainio, L., Tucker, M., & Ellis, R. (2007). Precision and power grip priming by observed grasping. *Brain and Cognition*, *65*(2), 195–207.

Wang, W., Degenhart, A. D., Collinger, J. L., Vinjamuri, R., Sudre, G. P., Adelson, P. D., … Weber, D. J. (2009). Human motor cortical activity recorded with Micro-ECoG electrodes, during individual finger movements. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (pp. 586–589).

Warrington, E. K., & Shallice, T. (1984). Category specific semantic impairments. *Brain, 107* (Pt 3), 829–854.

Westlake, K. P., & Nagarajan, S. S. (2011). Functional connectivity in relation to motor performance and recovery after stroke. *Frontiers in Systems Neuroscience*, *5*.

CURRICULUM VITAE