INVESTIGATION OF UNINTENTIONAL MOVEMENT IN PEOPLE WITH CEREBRAL PALSY TO IMPROVE COMPUTER TARGET ACQUISITION

by

Sara Marie Sibenaller

BS Biomedical Engineering, University of Rochester, 2006

Submitted to the Graduate Faculty of

The School of Health and Rehabilitation Science in partial fulfillment

of the requirements for the degree of

Masters of Science in Rehabilitation Science and Technology

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

THE SCHOOL OF HEALTH AND REHABILITATION SCIENCE

This thesis was presented

by

Sara Marie Sibenaller

It was defended on

July 22, 2008

and approved by

Rory A. Cooper, PhD, Distinguished Professor, Rehabilitation Science and Technology

Brad Dicianno, MD, Assistant Professor, Physical Medicine and Rehabilitation

Thesis Advisor: Dan Ding, PhD, Assistant Professor, Rehabilitation Science and Technology

INVESTIGATION OF UNINTENTIONAL MOVEMENT IN PEOPLE WITH CEREBRAL PALSY TO IMPROVE COMPUTER TARGET AQUISITION

Sara Marie Sibenaller, M.S.

University of Pittsburgh, 2008

People with Cerebral Palsy (CP) have difficulty using computer pointing devices due to unintentional movement in their upper extremities. Fifty percent of people with CP have impaired arm-hand function which limits their ability to interface with pointing devices and effectively control cursor movement on the computer screen. This thesis involves two studies which utilize an Isometric Joystick in order to access the computer and complete target acquisition tasks.

The first study titled "Quantification of Cursor Movement of People with Athetoid and Spastic Cerebral Palsy to Improve Target Acquisition," aims to guide real-time digital filter development for people with athetoid and spastic CP for target acquisition tasks. By investigating the cursor movement measures throughout the target acquisition trajectory we gained a better insight as to when and how to compensate for unintentional movement in people with CP. Results showed that both people with athetoid CP and spastic CP have more difficulty hovering over the target than they did moving to the target, indicating that filter development should focus on the hovering portion of the target acquisition task in order to improve target acquisition time.

The second study titled "Customized Control for People with Athetosis and Dystonia to Improve Computer Access," aims to develop a method to prescribe appropriate switch/scanning control for people with athetosis and dystonia as well as to determine if customized

switch/scanning control is more effective in completing icon selection tasks than the proportional isometric control. Results of this study suggest that switch/scanning control could be useful in moving on the most direct path to the target as shown by a significantly smaller percent distance error for customized control as compared to proportional isometric control ($F(1,6) = 361.2$, $p < 0.01$).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

There are many people who have guided me throughout my master's degree and have helped to make these studies a success. First, I would like to thank my committee, Dr. Ding, Dr. Dicianno, and Dr. Cooper for their guidance, support, and the many opportunities they have provided to me over the last two years. I would also like to thank the clinical coordinators Annmarie Kelleher, Emily Teodorski, and Michelle Tolerico for their help with participant recruitment, scheduling, and participant testing as well as the rest of the staff and students at the Human Engineering Research Laboratories who have been very supportive and kept me on track. I would especially like to thank Harshal Mahajan, Karl Brown, and Shiv Hiremath for their technical support and for help with participant testing, as well as Dr. Brewer and Christian Niyonkuru for help with statistical analysis.

Thank you everyone who participated in the studies as well as to the staff of the Center for Assistive Technology, United Cerebral Palsy, and Three Rivers Center for Independent Living who played a crucial role in participant recruitment and provided a place to conduct testing.

Lastly, I would like to thank my parents and the people most important to me who have supported my decisions and been there every step of the way and my friends both in Pittsburgh and throughout the United States who have always there for me.

1.0 INTRODUCTION

In the United States, there are estimated to be over 750,000 people who have Cerebral Palsy (CP). Worldwide, the number exceeds 17,000,000 [1]. For people with movement disorders associated with CP, everyday activities including computer usage range from difficult to impossible due to unintentional movement throughout the body. In order to understand the bigger picture of computer access for people with CP and the procedures of the studies discussed in this thesis, readers must first understand the definition of CP, classifications of CP, and current adaptive computer interfaces for people with CP.

1.1 CEREBRAL PALSY

Cerebral Palsy (CP), originally termed "Cerebral Paresis" by William Little in 1861, is a broad term used to describe a diverse group of syndromes that cause a non-progressive disorder. CP may be due to genetic diseases, cerebral dysgenesis, hypoxia-ischemia, stroke, and intrauterine exposure to infection or inflammation during prenatal or perinatal stages as well as post-natal injury due to infection, trauma, or stroke [2, 3]. This disability arises before or after birth to early in life in the cerebral region of the brain causing abnormal movements and postures throughout the body that persist throughout a person's life span [3, 4]. The incidence of CP worldwide is 2 to 3 per 1000 births. The risk is higher in low-birth-weight infants and in twin pregnancies. Improved survival rates in this population has lead to an increased prevalence of neurodevelopmental impairment [2, 3].

In 2005, a group of investigators set out to create a definition of CP that would provide a common language for researchers, clinicians, and health officials in order to improve communication between specialties. The Executive Committee for the Definition of CP presented this definition at the international symposium: "Cerebral Palsy describes a group of

disorders of the development of movement and posture, causing activity limitation, that are attributed to non-progressive disturbances that occurred in the developing fetal or infant brain. The motor disorders of cerebral palsy are often accompanied by disturbances of sensation, cognition, communication, perception, and/or behavior, and/or by a seizure disorder" [4]. This definition recognizes that there are several types of "motor disorders" associated with CP. These disorders can be classified into multiple groups including spasticity, athetosis, and dystonia. However, people do not always fall into only one category in which case they are diagnosed as having a mixed type of CP. For the purposes of this thesis, we will focus on the three classifications mentioned above.

Spasticity

Spasticity is a velocity-dependent increase in tonic stretch reflexes that are caused by abnormal processing of sensory afferent inputs to the spinal cord. It is characterized by increased resistance to externally imposed joint motion. Increased muscle stretch reflexes, occurrence of muscle spasms and clonus, weakness, and impairment of voluntary movements are all seen in addition to increased muscle tone [5]. Spasticity can be rated clinically with instruments such as the Penn Spasm Frequency Scale which asks participants to rate the number of spasms they have per hour [6] and using the Modified Ashworth Scale where clinicians grade the level of spasticity based on the assessment of resistance to passive movement at a joint [7].

Dystonia

Dystonia is a movement disorder that causes repetitive, patterned muscle contractions producing abnormal movements and postures in the trunk, neck, face, or extremities [5, 8]. Dystonia is a result of involuntary co-contraction of agonist and antagonist muscles. Dystonic postures can either be slow or rapid, change during different activities and positions, and become

fixed in some cases [8]. Dystonia can be scored clinically using the Global Dystonia Rating Scale [9].

Athetosis

Athetosis is a complex movement disorder frequently found in cases of CP. Athetoid motion is highly irregular and difficult to predict and it is said to lack fixed amplitude, rhythmicity, or direction [10]. Athetosis is typically described as slow, wormlike, writhing motions seen primarily in the upper extremities and mainly in the distal musculature and is seen in cases of CP [11-15]. Athetosis makes it difficult to perform coordinated tasks such as target acquisition on a computer screen because of the unintentional movement that appears in the upper limbs. Due to extreme variability in the movements associated with athetosis it is very difficult to quantify.

1.2 COMPUTER ACCESS FOR PEOPLE WITH CP

Computers are an integral part of life for work and personal applications. They are used to access the internet for communication, entertainment, and information dissemination including job and health information. A person's ability to use a computer greatly affects his or her ability to participate in society. Nearly 50 percent of people with CP have impaired arm-hand function and therefore have problems using the computer because of difficulties in grasping a pointing device, controlling cursor movement, and selecting on-screen features [16]. As a result, people who experience difficulty using pointing devices may not have access to information that could help them to have a better quality of life. Additionally, computers are increasingly used as a learning tool; therefore, students who are unable to use computers due to inadequate pointing and keyboard interfaces, as well as inadequate software control strategies, are at a disadvantage compared to their peers.

Computer access tasks that involve pointing and selecting operations with a pointing device include acquiring an icon on the desktop via clicking or hovering, dragging icons on the screen, scrolling up and down or left to right using a scroll bar, and maneuvering through drop-down menus. There are many types of pointing devices available for people with disabilities including people with CP. These devices include a traditional mouse, trackball mouse, and joystick that are usually moved by the hand or foot. Alternatively, a tongue touch mouse, a head-controlled mouse, head pointers, light pointers, and eye gaze are also commonly used by people with severe limb impairment [17].

Traditional Mouse

A traditional mouse is used for pointing, clicking, double-clicking, and dragging objects on a computer screen [18]. Traditional mice come in roller-ball and optical varieties and are an appropriate option for people with CP who have enough hand control to point and click, but are not appropriate if they have difficulty grasping and controlling cursor movement.

Trackball Mouse

A trackball mouse (Figure 1) is an inverted mouse with a ball mounted on a stationary base. There are buttons on the base used to select icons similar to a standard mouse. This control interface requires less range of motion because the ball is rotated by moving the hand or fingers over it. Therefore, it is an attractive option for people with CP who have decreased range of motion or reflex spasticity which can be triggered by joint motion. This type of mouse is available in a variety of sizes, shapes, and configurations [17].

Figure 1. Logitech Trackball Mouse [19]

Penny and Giles Joysticks

The Penny and Giles Joysticks come in either a trackball or proportional joystick (Figure 2). These input devices combine pointing control with a variety of buttons which control specific computer tasks. The buttons on the top of the joystick correspond to clicking, double-clicking, and dragging tasks and can be customized to fit the user. These additional buttons are advantageous to people with CP who have extensive unintentional movement or very little cursor control and cannot hover over an icon. The joystick also has ports in the back enabling other switches to interface with it for clicking, double-clicking, and dragging tasks.

Figure 2. Penny & Giles Trackball and Joystick Mouse [20]

Head Pointer

The head pointer (Figure 3) is designed to help people with limited arm-hand function but intact head control to use the computer and alternative augmentative communication devices by pointing to an icon on a keypad via gross head movements. This method is slow due to the accuracy needed to point to a specific icon.

Figure 3. Head Pointer [21]

Previous studies investigated the use of different pointing devices among people with disabilities including CP. Koester et al. developed computer skill assessment software in order to clinically evaluate computer skills among people with disabilities. This software allows clinicians and computer access specialists to quantify task completion time, cursor speed, and movement accuracy when using different computer pointing interfaces. It also helps clinicians make judgments about performance in order to select appropriate interventions [22].

Man and Wong evaluated two students with athetoid CP and their ability to use pointing devices to access a computer. Neither student had previous experience with computers due to the inability to use a traditional handheld pointing interface. Four different types of non-hand-operated computer access solutions were evaluated. Participants completed a questionnaire on the performance, comfort, and effort needed to use a pointing device and a target acquisition task based on the ISO 9241 Standards [23]. Participants in the study suggested evaluating the level of comfort with regard to duration of use and eye strain for the devices evaluated [24].

Although there are many types of adaptive pointing devices for individuals with disabilities, there has been very little research on quantitatively determining their usefulness for specific population such as people with CP. Despite these adaptive designs, there are still many people who are unable to use them due to the severity and frequency of their unintentional movement.

1.3 ISOMETRIC JOYSTICK

The two studies discussed in this thesis use an isometric joystick (IJ) as a computer pointing device. The IJ was developed by investigators at the Human Engineering Research Laboratories as an alternative control method for power wheelchair driving and computer access for people with movement disorders [25-27]. The IJ (Figure 4) was designed to be structurally similar to a traditional movement sensing joystick (MSJ). The internal structure of the joystick contains two strain gauges that are mounted orthogonally at the base of the handle. The voltage output from the joystick is proportional to the force exerted on the handle. The rigid handle has a dampening effect which is advantageous for people who have unintentional movement. Kimmich et al. found that people with upper extremity tremor showed a significant improvement when using the IJ as opposed to the MSJ during a virtual driving task with population of five people [28]. The IJ was also highly customizable for individual users with software-based algorithms [29].

Figure 4. IJ Developed at the Human Engineering Research Laboratories

The IJ has been deemed an acceptable control device by a number of studies on power wheelchair driving and computer access. Studies have shown that the IJ is comparable to the MSJ during real and virtual power wheelchair driving tasks using performance measures such as the root mean squared error of the trajectories and trial time [25, 26, 29]. Stewart et al. investigated the use of another type of IJ for people with CP during a computer target acquisition task and found that the MSJ was superior to the IJ in people with previous power wheelchair driving experience when looking at the movement time and the ability to acquire a target via hovering. However, for people without power wheelchair driving experience, the IJ performed the same as the MSJ [30]. Rao et al. found that the average movement time for a set of target acquisition tasks was longer using the IJ than the MSJ among 14 participants with CP who had no prior experience in using a joystick [31]. It is important to note that the IJ used in the study was not customized to individual users and its sensitivity could affect the participants' performance.

1.4 ORGANIZATION OF THE THESIS

This thesis discusses two studies aimed at improving computer target acquisition for people with movement disorders associated with CP. Both studies involve using the HERL IJ to access the computer and complete a sequence of target acquisition tasks. The first study titled "Quantification of Cursor Movements for People with Athetoid and Spastic Cerebral Palsy during Target Acquisition" investigates the characteristics of cursor movements in people with athetoid and spastic CP during target acquisition tasks with the goal of guiding software-based algorithms or new control interfaces to improve computer interaction for this population. The second study titled "Customized Control for People with Athetosis and Dystonia" investigates customized switch and scanning control based on the directional and graded force produced by people with athetosis and dystonia. Both of these studies are part of larger research efforts. Their protocols and results will be discussed in sections 2.0 and 3.0 respectively.

2.0 QUANTIFICATION OF CURSOR MOVEMENT FOR PEOPLE WITH ATHETOID AND SPASTIC CEREBRAL PALSY DURING TARGET ACQUISITION

2.1 INTRODUCTION

Computers are increasingly used by individuals with disabilities to communicate and participate in society [32]. With the popularization of computer applications using graphical user interfaces (GUI), users need to use pointing devices such as a mouse or a joystick to reach a position on the screen. People with athetoid CP usually have difficulty operating these pointing devices due to unintentional movements and associated spasticity [3]. People with spastic CP also have difficulty due to jerky and unpredictable movements caused by increased tone and permanently contracted joints [3].

One of the reasons why people with CP have difficulty using traditional pointing devices is that they have to move the device in order to produce cursor movement. In this study, an isometric joystick (IJ) with a rigid force-sensing handle developed at the Human Engineering Research Laboratories was used to allow people with CP to control cursor movement by exerting force with very little hand or arm movement. In addition, the IJ was designed to accept any input force level the user can generate and provide a wide dynamic range of inputs, which overcomes the problem of saturation of conventional MSJs where an involuntary movement might easily drive the conventional joystick to its limiting position.

Previous studies have been performed to gain a better understanding of human computer interaction for target acquisition among people with impaired motion using various pointing devices and to design interventions to improve the effectiveness of pointing control. The act of pointing and selecting targets on the screen depends on the speed and accuracy of the user's

movement. Performance measures have been developed and discussed in previous studies to evaluate different pointing devices and understand user behaviors.

MacKenzie et al. proposed seven accuracy measures including target re-entry, task axis crossing, movement direction change, orthogonal direction change, movement variability, movement error, and movement offset. Four pointing devices including a mouse, trackball, joystick, and touchpad were evaluated with 12 able-bodied participants using the cursor measures listed above. It was found that these measures gave information about cursor pointing tasks beyond the traditional measures of speed and accuracy [33].

Keates et al. studied how force feedback affects computer interaction for five participants with athetoid and/or ataxic CP. Point and click tasks were analyzed using Fitts' Law and the seven cursor measures were proposed. The results showed significant improvement in performance for all users with force-feedback and the seven cursor measures offered insight as to how performance improves [34].

To gain a better understanding of impaired movement, Hwang et al. investigated the cursor trajectories of six people with CP and three able-bodied users during a point and click task using a submovement analysis. The results showed that some motion-impaired users pause more often and for longer than able-bodied users and require up to five times more submovements to complete the same task [35].

Radwin et al. studied ten participants with no movement disabilities and two with CP during target acquisition tasks using a conventional mouse and a lightweight ultrasonic head-controlled computer pointing device. Performance measures including movement time, cursor path distance, and root-mean-square were used to evaluate their performance [36].

Sibenaller et al. used kinematic and performance measures derived from Fitts' Law to quantify the cursor movements of seven participants with athetoid CP in a target acquisition task and found that participants with athetoid CP had higher average peak acceleration and lower index of performance than that of an able-bodied participant [37].

Most recently, efforts have been made to develop filters to smooth cursor trajectory and decrease overall cursor movement time for people with CP during target acquisition tasks. Lopez et al. designed a non-linear filter based on a cascade-correlation neural network model and Olds et al. used an auto regressive stretching average method to filter the athetoid movement during target acquisition tasks. Both studies reported improvement in target acquisition time in offline experiments, but did not test the algorithms in real time with individuals who have CP [38, 39]. Attempts have also been made to replicate athetoid movement in order to facilitate the development of assistive computer interfaces for people with movement disorders [39, 40].

The primary objective of this study was to investigate the characteristics of cursor movements during a sequence of target acquisition tasks using an IJ by people with athetoid and spastic CP. The study used a set of comprehensive cursor measures to evaluate the performance of 15 individuals with athetoid and spastic CP, and 10 individuals with spastic CP only. The information collected in this study will provide a better understanding of impaired movements in people with CP and insights on optimizing, enhancing, and developing new methods to improve human-computer interaction among this population. There were three specific aims in this study including:

1. To investigate the characteristics of cursor movements including basic movement measures, Fitts' Law measures, submovement measures, and kinematic measures during target acquisition tasks in people with athetoid and spastic CP.

2. To investigate the difference in cursor movements between people with athetoid CP and people with spastic CP.
3. To determine the relationship between the characteristics of cursor movements and clinical measures among people with athetoid and spastic CP.

2.2 METHODOLOGY

2.2.1 Participant Recruitment and Inclusion Criteria

A total of 25 people with CP were recruited to participate in this study including 15 people with athetoid CP and 10 people with spastic CP who matched people with athetoid CP by gender and age +/- 5 years. Participants were recruited through the Human Engineering Research Laboratories wheelchair user's registry, with the help of local clinicians, and by networking with local disability organizations and professionals. The inclusion and exclusion criteria for this study were:

1. Must be over the age of 18.
2. Must have CP.
3. Must be able to sit for at least 3 hours.
4. Must not have any active pressure wounds (could be worsened by prolonged sitting)
5. Must not have had a seizure in the past 90 days.

2.2.2 Experimental Setup

The experimental setup for this study consisted of an IJ that was secured to the table via a mounting bracket. The IJ was connected to a laptop computer via a serial cable. The participants were asked to position themselves so that the IJ was in the same position as their movement sensing joystick (Figure 5). A height adjustable table was used in order to best

accommodate the participants when necessary. The laptop computer was placed at a comfortable distance away from the participants as they sat in their own wheelchairs. If the person did not use a wheelchair, they were given a chair with an arm rest to sit in during the study.

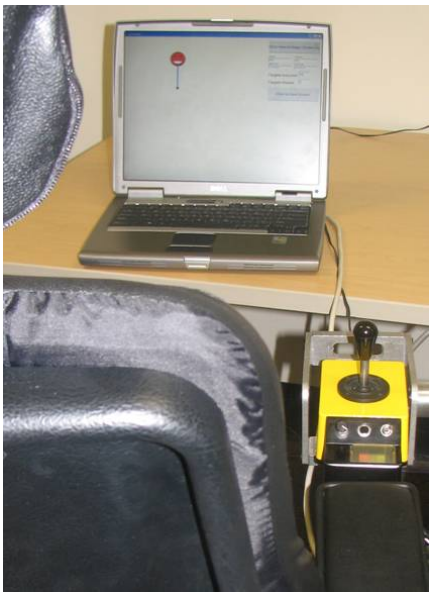

Figure 5. The Experimental Setup

Participants were given an option to select from a variety of joystick handles (Figure 6). This was a key factor in allowing the participant to interact with the joystick and subsequently control the cursor on the computer screen. The handle was matched as best as possible to the participant's current handle on his or her wheelchair.

Figure 6. IJ Handles

2.2.3 Protocol

The protocol was reviewed and approved by the Institutional Review Boards at the University of Pittsburgh and Carnegie Mellon University. The nature of the study was explained and written informed consent was obtained from all participants before data collection. All participants underwent a brief upper limb neurological examination and were evaluated using the Modified Ashworth Scale (MAS) (Appendix A) on the wrist and elbow joints of both upper limbs in flexion and extension. Information on demographics including age, sex, movement disorder classification, type of wheelchair used, and computer usage including computer independence, computer usage frequency, and type of pointing device used were collected. The Barthel Index Score and Penn Spasm Frequency Scale (Appendix A) were also obtained. Median or mean scores were obtained for each individual for ordinal and continuous scales, respectively, and then overall median or mean scores were calculated for each subject group.

The test session started with a 10-minute session where the participants practiced moving a cursor on a blank screen to become familiar with the IJ. Next, they underwent a screening task shown in Figure 7. Each participant was asked to move the cursor to the target (diameter of approximately one inch) using the IJ. The target is considered to be successfully acquired when the cursor entered the target area and remained inside for at least one second, as indicated by a red smiley face shown in Figure 7. A trial was automatically terminated and recorded as a failure, as indicated by a green frown face (Figure 7) if the target had not been acquired within 20 seconds. At least one out of the ten trials needed to be successfully completed in order for the participant to be eligible to participate in the study. These minimal criteria ensured that the participant was able to manipulate a joystick and produce movements, yet did not exclude participants who may not be able to control cursor movement.

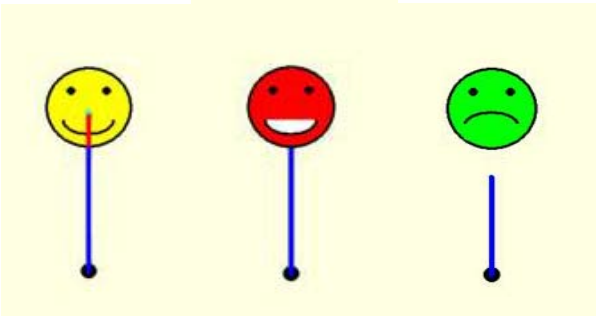

Figure 7. The Screening Target Acquisition Trials: Before Acquisition (Left), Successful trial (Middle), and Unsuccessful trial (Right)

After passing the screening task, each participant was asked to perform 100 multi-directional target acquisition trials in sets of 10. Targets appeared on the screen in random locations. These trials consisted of moving a cursor from a blue target to a yellow target with a diameter of approximately one inch (Figure 8). The participant had to dwell inside the target for at least 2 seconds to complete the trial successfully. A trial was recorded as a failure if the target had not been acquired within 20 seconds. Rest was given in between sets of 10 trials when needed and at the researcher's discretion. The cursor positions, digital output signals from the IJ, and the trial status (success or failure) were recorded using customized software (Figure 8) written in Borland C++ 5.0 (Appendix C) [41].

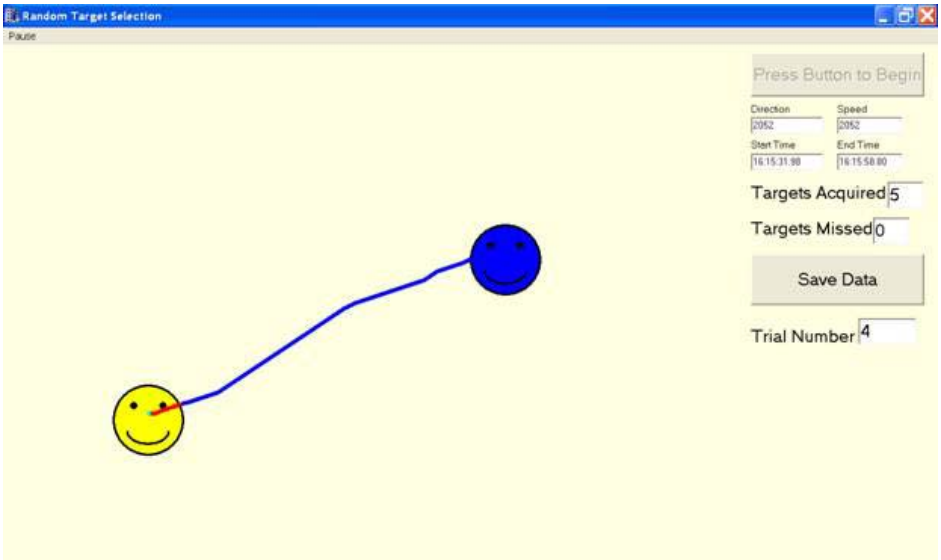

Figure 8. Random Target Acquisition Task

2.2.4 Data Reduction

The overall Modified Ashworth Scale (MAS) used in the data analysis was obtained by taking the median of the four MAS scores for the wrist and elbow joints in flexion and extension in the upper extremity that controlled the joystick. Cursor movement data were sampled at a rate of 100 Hz and down sampled to 20 Hz for post processing. The following cursor movement measures were calculated using MATLAB and averaged over 100 trials [42].

Basic Movement Measures

- *Percentage of Acquired Targets*: the number of acquired targets divided by the number of attempted targets multiplied by 100.
- *Actual Distance (AD)*: the direct distance in pixels between the cursor starting point and the center of the target.
- *Movement Time (MT)*: the time in seconds needed to acquire a target. Participants were given a maximum of 20 seconds to complete the trial successfully. This measure was normalized by the actual distance, as the targets were placed at random distances apart from each other.
- *Reaction Time:* the time for the participant to initiate cursor movement and mathematically defined as the amount of time before cursor velocity is greater than zero.
- *Total Distance (TD)*: the length of cursor trajectory in pixels from the starting point to the center of the target normalized by the actual distance.
- *Target Width*: the diameter of the circular target. It was set to 96 pixels or approximately 1 inch wide.

- *Percent Distance Error*: the ratio of the difference between total distance (TD) and actual distance (AD) over the total distance (TD) as shown below.

$$PDE = \frac{|AD - TD|}{TD} * 100$$

Fitts' Law Measures

Measures of Fitts' Law define the properties of human movement during target acquisition. Fitts' Law has been used to quantify the movements of people with CP and evaluate different computer pointing devices [26, 34, 35, 43].

- *Index of Difficulty (ID)*: the difficulty of a motor task defined as shown below.

$$ID = \log_2\left(\frac{AD * TW}{TW}\right)$$

- *Index of Performance (IP)*: the rate of information transmission computed in bits per second defined as shown below.

$$IP = \frac{ID}{MT}$$

Submovement Measures

A submovement was considered to begin at the end of a pause when the cursor velocity was greater than zero and end at the beginning of next pause when cursor velocity dropped to zero. A pause was defined as a period when the velocity remains zero for at least one data point.

- *Mean Pause Duration:* the mean duration for a pause averaged for all pauses within a trial.
- *Number of Pauses*: a count of the number of pauses in a trial including the dwell pause at the end of the trial, but excluding the reaction pause in the beginning of the trial.

- *Mean Submovement Duration*: the average length of time that a submovement lasts averaged for all submovements within a trial.
- *Number of Submovements*: a count of the number of submovements in a trial. Based on the definition of a pause and a submovement, the number of submovements should be approximately equal to the number of pauses.
- *Percentage of Submovements Before First Target Entry*: the percentage of submovements before the first target entry including the movement that helped place the cursor in the target.
- *Percentage of Submovements After First Target Entry*: the percentage of submovements after initial target entry excluding the movement that helped place the cursor in the target for the first time. In a perfect trial, there would be no submovements after the first target entry.
- *Percentage of Movement Time Before First Target Entry*: the percentage of the time for the cursor to move towards the target as compared to the total movement time in a trial.
- *Percentage of Movement Time After First Target Entry*: the percentage of time after the cursor moves in the target for the first time until the target is acquired or the trials ends as compared to the total movement time in a trial.
- *Number of Slip-offs*: the number of times the cursor moves outside the target after first target entry.

Kinematic Measures

- *Average Velocity*: the mean instantaneous velocity for a trial.
- *Average Peak Acceleration*: the mean of the maximum absolute acceleration for a trial.

- *Number of Acceleration Peaks and Valleys*: the number of acceleration peaks and valleys within a trial.

2.2.5 Statistical Analysis

Exploratory data analysis preceded all formal statistical analysis. Descriptive statistics and frequencies were calculated for cursor movement measures, clinical measures including Penn Spasm Frequency, Barthel Index, and Modified Ashworth Scale, demographics, and computer usage measures obtained from the questionnaire.

A Mann-Whitney U Test was used to compare the cursor movement measures between the participants with athetoid CP and those with spastic CP. As this was a preliminary study, a Bonferroni correction was not used to adjust the significance level. A multivariate test was considered but not used. The assumptions of the test, including even groups and a large sample size, were not met by the data set.

Spearman Rho correlations were calculated to determine significant relationships between cursor movement measures and clinical measures. All statistical analysis was completed using SPSS version 14.0 software [44]. Statistical significance was set at 0.05.

2.3 RESULTS

Demographics

Ten women and fifteen men with an average age of 42.0 ± 11.7 years were enrolled in the study. Fifteen people had athetoid and spastic CP, and 10 had spastic CP only. Twenty-two people used a power wheelchair, one person used a scooter, one person used a manual wheelchair and did not self-propel, and one person ambulated. The demographic information of each diagnostic group is shown in Table 1.

Measures	No. of Participants (%) Athetoid (n=15)	Spastic (n=10)
Gender		
Male	9 (60)	6 (60)
Female	6 (40)	4 (40)
Mobility Device		
Power Wheelchair	14 (93)	8 (80)
Scooter	0 (0)	1(10)
Manual Wheelchair (no self propulsion)	0 (0)	1(10)
Ambulator	1 (7)	0 (0)

Table 1. Demographics in the Two Diagnosis Groups

Computer Usage

Computer usage, specific task frequency, and type of pointing device are shown in Table 2. The types of pointing devices used by people who use a computer were divided into three categories including standard hand-operated mouse, trackball mouse, and other type of pointing device including Penny & Giles joystick, Penny & Giles trackball, head pointer, and communication device that interfaces with the computer.

Measures	No. of Participants (%) Athetoid (n=15)	Spastic (n=10)
Computer Independence		
Independent	13 (87)	10 (100)
Dependent	2 (13)	0 (0)
Computer Usage Frequency		
Never	4 (27)	0 (0)
Daily	7 (47)	8 (80)
Several Times a Week	4 (27)	2 (20)
Computer Pointing Device		
Traditional Mouse	3 (20)	6 (60)
Trackball Mouse	5 (33)	3 (30)
Other Type of Pointing Devices	3 (20)	1 (10)
Do Not Use a Computer	4 (27)	0 (0)

Table 2. Computer Usage Frequencies

Cursor Movement Measures

Table 3 shows the cursor movement measures in four categories including basic measures, submovement measures, Fitts' Law measures, and kinematic measures.

Cursor Movement Measures	Mean Value ± Standard Deviation: Athetoid CP (n=15)	Mean Value ± Standard Deviation: Spastic CP (n=10)	p-value
Basic			
Percentage of Acquired Targets (%)	84.7 ± 26.4	88.8 ± 28.5	0.37
Actual Distance (pixels)	293.2 ± 92.8	301.6 ± 92.5	0.89
Movement Time (seconds)	9.3 ± 4.8	8.3 ± 4.5	0.57
Movement Time (normalized)	0.07 ± 0.09	0.06 ± 0.08	0.43
Reaction Time (seconds)	0.9 ± 0.9	1.3 ± 0.7	0.22
Total Distance (Normalized)	9.8 ± 17.8	3.9 ± 6.1	0.09
Percent Distance Error (%)*	**53.7 ± 24.1**	**36.0 ± 25.0**	**0.04**
Submovement			
Mean Pause Duration (seconds)*	**0.6 ± 0.3**	**1.0 ± 0.4**	**0.01**
Number of Pauses (#)*	**7.8 ± 5.6**	**5.6 ± 8.6**	**0.02**
Number of Pauses (Normalized)	0.06 ± 0.07	0.04 ± 0.05	0.07
Mean Submovement Duration (seconds)*	**0.9 ± 0.4**	**1.5 ± 0.5**	**0.01**
Number of Submovements (#)*	**7.8 ± 5.6**	**5.6 ± 8.6**	**0.02**
Number of Submovements (Normalized)	0.06 ± 0.07	0.04 ± 0.05	0.07
Percentage of Submovements Before First Target Entry (%)	49.2 ± 20.6	58.7 ± 21.3	0.22
Percentage of Submovements After First Target Entry (%)	50.8 ± 20.6	41.3 ± 21.3	0.22
Number of Target Slip offs After First Target Entry (#)*	**1.9 ± 1.6**	**0.6 ± 0.7**	**0.01**
Percentage of Movement Time Before Target Entry (%)	36.1 ± 17.3	39.3 ± 17.4	0.61
Percentage of Movement Time After Target Entry (%)	63.9 ± 17.3	60.6 ± 17.3	0.61
Fitts' Law			
Index of Difficulty	7.7 ± 1.0	7.8 ± 1.0	0.93
Index of Performance	1.2 ± 0.5	1.2 ± 0.4	0.76
Kinematic			
Average Velocity (pixels / second)*	**107.1 ± 52.1**	**63.5 ± 11.1**	**0.01**
Mean Peak Acceleration (pixels / second^2)	89.7 ± 47.9	83.9 ± 50.8	0.68
Number of Acceleration Peaks and Valleys (Normalized)	0.5 ± 0.9	0.3 ± 0.4	0.10

*Significant difference existed between the two groups at an alpha level of 0.05

Table 3. Descriptive Statistics of Cursor Movement Measures

Significant differences were found between people with athetoid CP and those with spastic CP. People with athetoid CP had a significantly larger percent distance error than people

with spastic CP (Z = -2.052, p = 0.04). Additionally, people with athetoid CP had a significantly higher number of submovements and pauses than people with spastic CP. However, when normalized by the actual distance there was no significant difference between the groups. There was, however, a trend that indicated that people with athetoid CP had a larger number of submovements (p = 0.07) and pauses (p = 0.07) than people with spastic CP. People with athetoid CP also had significantly shorter pause durations (Z = -2.607, p = 0.01) and significantly lower mean submovement durations than people with spastic CP (Z = -2.496, p = 0.01). Lastly, people with athetoid CP had a significantly higher number of target slip offs after first target entry (Z = -2.415, p = 0.01) as well as a significantly higher average velocity (Z = -2.552, p = 0.01) than people with spastic CP.

Clinical Measures

Table 4 shows the mean clinical measures for both groups. The range of scores for the Barthel Index is 0-100 with higher scores indicating greater independence in activities of daily living. The range of scores for the Modified Ashworth Scale is 0-5 with lower scores indicating lower levels of spasticity. Similarly, the range of scores for the Penn Spasm Frequency scale is 0-5 with lower scores indicating less frequent spasticity.

	Scores					
Measure (n=8)	Athetoid			Spastic		
Median Modified Ashworth Scale (range)	2		(1-4)	4		(2-4)
Mean Barthel Index (SD)	41.7	±	(28.2)	57.5	±	(28.9)
Median Penn Spasm Frequency (range)	4		(2-5)	3		(2-3)

Table 4. Descriptive Statistics of Clinical Measurement Measures

Correlations between Clinical Measures and Cursor Movement Measures

There were several significant correlations found between the cursor movement measures and clinical measurement scores. There was a significant negative correlation between Modified Ashworth Scale and mean submovement duration ($p < 0.01$, $R^2 = 0.58$) for the athetoid group, $p < 0.01$, $R^2 = 0.75$ for the spastic group), indicating that people with a higher level of spasticity had shorter submovements. For people with spastic CP there was a significant negative correlation between the Penn Spasm Frequency score and the percentage of acquired targets ($p=0.03$, $R^2 = 0.48$) indicating that people with more frequent spasms acquired a smaller percentage of targets. There was also a significant positive correlation for people with spastic CP between the Penn Spasm Frequency Scale and the number of submovements after first target entry ($p = 0.03$, $R^2 = 0.46$) indicating that people with spastic CP with more frequent spasms have a larger number of submovements after first target entry.

2.4 DISCUSSION

While previous target acquisition tasks used a circular layout of targets, the target acquisition tasks in this study included placing targets at random distances in random directions, which increased the difficulty of the task. However, this caused problems when comparing the performance between the participants with athetoid CP and those with spastic CP. Though the index of difficulty of the trials were similar between the two groups (7.7 ± 1.0 for the athetoid group and 7.8 ± 1.0 for the spastic group), we normalized some of the cursor movement measures by the actual distance in each trial for the purpose of a fair comparison.

Previous studies showed that able-bodied people typically have one submovement with a symmetric bell-shaped velocity profile, while people with impaired motion such as people who had a stroke usually have more submovements and irregular velocity profiles with multiple peaks

during tasks that require rapid hand movements [45, 46]. Our results showed that participants with athetoid CP in this study had an average of 7.8 pauses and submovements per trial which was higher than those with spastic CP who had 5.6 pauses and submovements per trial. Both groups had more submovements and pauses than necessary to complete a target acquisition trial. This result is not surprising considering the unintentional movement in people with athetoid CP and tonic stretch reflex in people with spastic CP. The trend that participants with athetoid CP have more pauses and submovements than those with spastic CP ($p=0.07$ for the normalized number of pauses and submovements) indicates that the athetoid CP group requires more breaks to complete a target acquisition task.

Additionally, the duration of the submovements was significantly higher for people with spastic CP (1.5 seconds) than people with athetoid CP (0.9 seconds) ($p=0.01$). Likewise, the pause duration was significantly higher for people with spastic CP (1.0 seconds) as compared to people with athetoid CP (0.6 seconds) ($p = 0.01$). Longer pause durations and submovement durations in people with spastic CP as compared to people with athetoid CP is due to having less pauses and submovements within a trial. Therefore the pauses and submovements of people with spastic CP last longer because the movement is more controlled as compared to people with athetoid CP who have more choppy movement. Cursor movement efficiency could be increased by increasing the length of the submovements and decreasing the length of the pauses. Decreasing the number of pauses and submovements would also increase the efficiency of the movement trajectory by decreasing the overall movement time.

The percent of movement time before and after the first target entry showed that both the athetoid CP group and the spastic CP group spent over 60% of their time acquiring the target as compared to about 40% of their time moving towards the target. Additionally, the number of

target slip-offs after first target entry was found to be significantly higher for people with athetoid CP than with spastic CP ($p = 0.01$). These results suggest that filter development efforts should focus on the cursor movement once it is near the target during the hovering task. Hwang et al. concurs with the need to investigate movements around the target. They showed that some of the motion-impaired users in their study exhibit verification times (the time from the end of the last submovement to the end of the trial) that were 65% of the average time required for the able-bodied users to fully complete the task [35].

There are many possibilities for improving target acquisition via hovering including modifying the target properties as opposed to the cursor movement. The duration of the mean submovements and pauses for both groups was less than the hovering time needed to acquire the target indicating that hovering time may need to be decreased in order to increase a person's ability to acquire a target. The ideal hovering time for each group may be different since the submovement durations and pause durations were significantly different for each group. Therefore, it may be beneficial to customize hovering time based on a person's specific submovement and pause durations. Further research is needed to understand the ideal relationship between hovering time and submovement and pause durations for people with CP.

Other possibilities for changing the properties of the targets on the screen include increasing the target area so there is a larger target to hover on. This width could be optimized for each individual based on the average position of the cursor on the screen after first target entry. Another approach would be to attach gravitational pull or magnetism to the target. Therefore, when a cursor comes close to the target it would be pulled into the center [47].

Percent distance error was significantly higher for people with athetoid CP than spastic CP ($p=0.04$) meaning that people with spastic CP move with a more direct path to the target than

people with athetoid CP. Difficulty maintaining a straight path may also be due to the jerky movements as seen by the trend that people with athetoid CP have a larger number of acceleration peaks and valleys than people with spastic CP ($p = 0.10$) and could be compounded by high cursor movement velocities which indicate difficulty with controlling cursor movement.

The average cursor movement velocity was significantly higher for people with athetoid CP than those with spastic CP ($p = 0.01$). High velocity can make it difficult to move to a target if it is close and dwell on a target without overshooting or slipping off it as discussed earlier. A velocity dependent filter might be useful for people with athetoid CP in order to slow down their cursor movements when they are too jerky or too fast to control. In doing so, it is important to optimize the relationship between the velocity of the cursor and the time it takes to reach the target. If the velocity of the cursor is made to be too slow the users will become frustrated, so a balance between cursor speed and movement time is crucial to design an effective filter of this type. More individualized customization may be needed to improve cursor movement and target acquisition in this population.

It is noted that a submovement in this study was defined as cursor motion between two zero velocity moments, however, several velocity peaks could exist within one submovement [45]. People with athetoid CP usually have difficulty pausing due to the constant motion of their upper limbs. Therefore, it may be beneficial to define submovements by velocity peaks to gain a further understanding of their movement patterns.

Clinical measures including Modified Ashworth Scale and Penn Spasm Frequency Scale are useful in measuring the frequency and level of spasticity from day to day. Results showed that people with athetoid CP and people with spastic CP with higher MAS scores had shorter submovements ($p < 0.01$, $R^2 = 0.58$ for the athetoid group and $p < 0.01$, $R^2 = 0.75$ for the spastic

group). Results also show that people with spastic CP with higher Penn Spasm Frequency scores acquired a smaller percentage of targets ($p=0.03$, $R^2 = 0.48$) and had more submovements after first target entry ($p = 0.03$, $R^2 = 0.46$). These results indicate that MAS score for people with athetoid CP and spastic CP and Penn Spasm Frequency Score for people with spastic CP relate in some way to a person's performance in target acquisition tasks. Because unintentional movement can increase or decrease from day to day based on changes in medication, medical comorbidities, mood, or for other reasons these scores should be collected during each subsequent research visit. These clinical measures can give us insight into a person's physical state on any given day and may be used as a tool to explain why a person performs better on one day than another. Knowing that a person's needs change from day to day also helps researchers to design software that can be adjusted to the person's ability on a particular day.

Lastly, a person's current technology and how they use it should be considered when designing a new type of device or control method. In this study, only 73% of people with athetoid CP used computers as compared to 100% of people with spastic CP. Over half of the participants with spastic CP (60%) use a traditional hand operated mouse as opposed to only 27% of participants with athetoid CP. Alternatively, 20% of participants with athetoid CP used a more advanced type of pointing devices which provided an alternative control method as opposed to 10% of participants with spastic CP. Thus, the need for more alternative control methods for people with athetoid CP is paramount.

2.5 CONCLUSION

The results of this study show that both people with athetoid and spastic CP have difficulty completing target acquisition tasks. People in both groups had more submovements than necessary and did not take the most direct path to the target. They also struggled more with hovering over the target than they did moving to the target. People with athetoid CP had lower performance than people with spastic CP as indicated by the cursor movement measures. This study provides insights on optimizing, enhancing, and developing new methods to improve computer access in this population. Potential improvements include changing the properties of the targets, customizing interface parameters, and optimizing the movement time and cursor trajectory when designing a filter.

3.0 CUSTOMIZED CONTROL FOR PEOPLE WITH ATHETOSIS AND DYSTONIA TO IMPROVE COMPUTER ACCESS

3.1 INTRODUCTION

Athetosis and dystonia are movement disorders associated with CP. Athetosis is described as involuntary, wormlike, writhing motions seen primarily in the upper extremities and mainly in the distal musculature [12-15]. Dystonia is described as repetitive, patterned muscle contractions producing abnormal movements and postures in the trunk, neck, face, or extremities [5, 8]. People with athetosis and dystonia have difficulty using proportional control computer pointing devices such a traditional mice, joysticks, trackballs, and head-mounted pointers due to the unintentional movement throughout the body which makes it difficult to grasp the device and control the cursor movement on the screen [25].

Therefore, switch control is sometimes used for computer access for people who have difficulty or cannot use a pointing device. Despite advances in switch and scanning technology, especially in the field of Alternative Augmentative Communication (AAC) devices, proficient use of these systems depends on the user's ability to activate and deactivate the device quickly and repetitively by pressing the buttons or switches which can be difficult due to impairment. In this study, switch and scanning is used in conjunction with tuning software in order to customize the IJ to the person's needs according to their ability to produce directional and graded forces.

Dystonia and athetosis create abnormal postures and coordination, increased tone, and involuntary movements [4, 48, 49] that may limit the use of current switch and scanning devices. However, literature suggests that people usually have at least two available directions of movement [49-51]. Thus, a customizable switch control interface that requires only simple graded or directional movements could provide better controls for this population.

The primary objective of this study was to develop a customized switch control interface using an IJ and compare it with the proportional isometric control among people with dystonia and athetosis. The specific aims of this study include:

1. Determine if people with athetosis and dystonia perform better with customized switch control than with proportional isometric control during computer access tasks.
2. Determine if there is a difference in the percentage of acquired targets, total time, reaction time, and percent distance error between people with athetosis and people with dystonia when using the customized switch control and proportional isometric control for computer access.

3.2 METHODOLOGY

3.2.1 Participant Recruitment and Inclusion Criteria

A total of 11 people with athetoid and dystonic CP participated in this study. Participants were recruited through the Human Engineering Research Laboratories wheelchair user's registry, with the help of local clinicians, and by networking with local disability organizations and professionals. The inclusion /exclusion criteria for this study were:

1. Must be over the age of 18.
2. Must have either athetosis or dystonia.
3. Must be able to sit for at least 3 hours.
4. Must not have any active pressure wounds (could be worsened by prolonged sitting)
5. Must not have had a seizure in the past 90 days.

3.2.2 Experimental Setup

The physical experimental setup was discussed in section 2.2.2. The software setup consisted of two interfaces including the joystick tuning interface and target acquisition

interface. The joystick tuning interface (Figure 9) was used to customize dead zone, bias axis, template settings, and gain settings of the IJ to individual users [52]. The target acquisition interface (Figure 10) was used to collect information on the participant's ability to generate directional and graded forces.

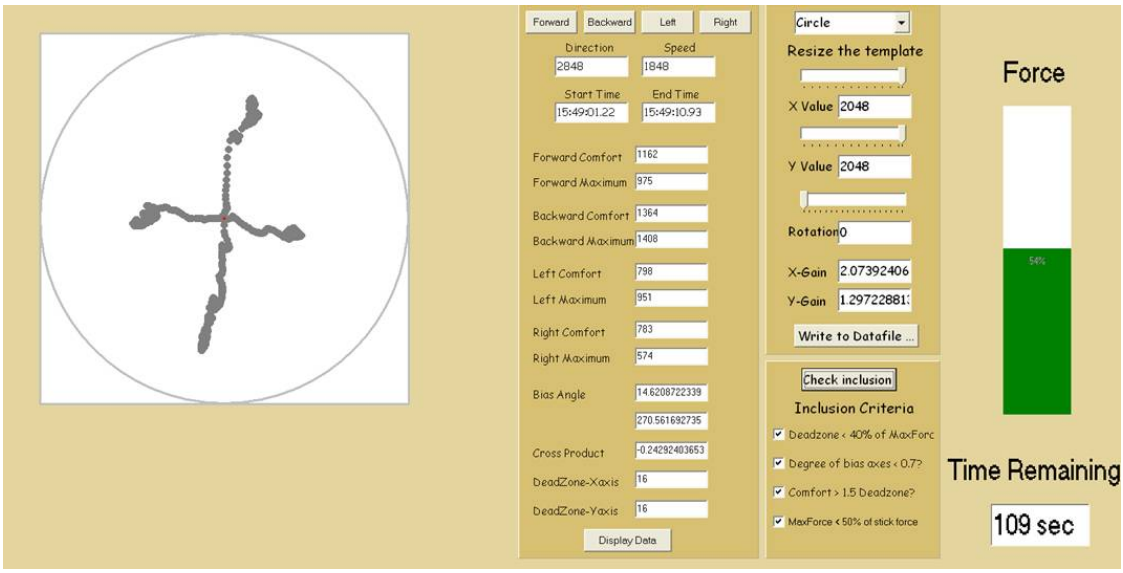

Figure 9. Tuning Interface for IJ developed at HERL

3.2.3 Protocol

The protocol was reviewed and approved by the University of Pittsburgh and the Veterans Administration Pittsburgh Healthcare System Institutional Review Boards. The nature of the study was explained and written informed consent was obtained from all participants before data collection. This study consisted of two visits.

Visit 1

All participants underwent a brief upper limb neurological examination and were evaluated with the Modified Ashworth Scale (MAS) at the wrist and elbow joints of both upper extremities in flexion and extension. Information on demographics including age, sex, movement disorder classification, type of wheelchair used, and computer usage including computer

independence, computer usage frequency, type of pointing device used, and task specific internet usage frequency were collected. The Barthel Index Score, Penn Spasm Frequency Scale, and Global Dystonia Rating Scale (see Appendix A) were also obtained.

First, the IJ was tuned to the participant using the joystick tuning interface. Next, participants were given the opportunity to practice ten target acquisition tasks using the joystick. After practice, participants were presented with the target acquisition interface where they were asked to move the cursor as quickly and accurately as possible from the center starting point to the target when it appeared on the screen and hover on the target for two seconds to successfully complete the task. During the trial, 32 targets appeared on the screen in 8 different locations (4 times in each position). Each target location was associated with one graded force and one direction, and appeared on the screen for 10 seconds and then disappeared. If the participant was able to successfully acquire a target at least 75% of the time, he/she was considered to be able to produce the direction/force combination and the information was used to design a customized switch control algorithm.

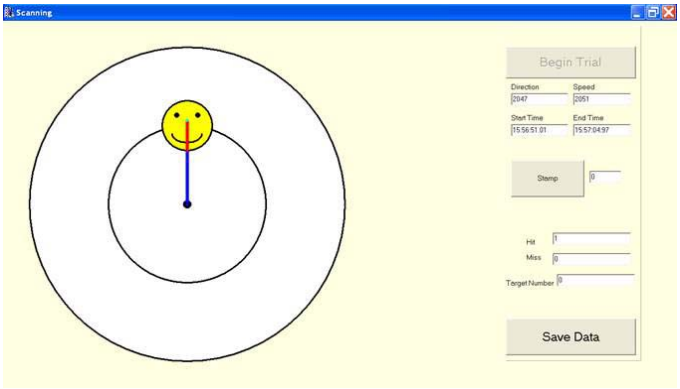

Figure 10. Target Acquisition Interface

Prior to Visit 2

Based on the combination of directional and graded forces produced by the participant in Visit 1, a customized switch control was designed where the IJ was converted into a switch that

could interface with scanning control or act as a direct acquisition device. For example, if a participant could only exert one graded force in one direction, single-switch scanning would be prescribed where he/she could select a movement direction (forward, backward, left, and right) from a scanning menu by applying a force on the joystick when the appropriate direction is highlighted on the menu (see Figure 11 for example of scanning menu). The cursor would then begin to move in the selected direction and scanning would be paused. In order to stop cursor movement, the participant would apply another force at which time cursor motion would stop and scanning would continue.

If the participant had the ability to produce two graded or directional forces, he/she could use one graded or directional force to scan through the menu options and the other graded or directional force to select the movement option. There were many possibilities for switch scanning control in this study, all of which were customized to the individual participant.

Figure 11. On Screen Scanning Interface

Visit 2

After the customized switch control was designed for the individual, the target acquisition interface was used to evaluate it and compare it with the proportional isometric control. Since there was generally a few months between visit 1 and 2, the IJ was tuned again at the beginning of the second visit. Participants were then coached on how to use the customized switch control and given the opportunity to practice using the IJ with the customized control in

15 target acquisition tasks. After practice, participants were presented with the target control interface where they were asked to acquire a set of 32 targets using the customized switch control and the proportional isometric control, respectively.

3.2.4 Data Reduction

Movement data was sampled at 100 Hz and down sampled to 20 Hz for post processing. The following cursor movement measures were used to compare the customized switch control with the proportional isometric control.

- *Percentage of Acquired Targets*: the number of acquired targets divided by the number of attempted targets multiplied by 100.
- *Movement Time*: the time in seconds needed to acquire a target. Participants were given a maximum of 20 seconds to complete the trial successfully.
- *Reaction Time:* the time for the participant to initiate cursor movement. Reaction time is mathematically defined as the amount of time before cursor velocity is greater than zero.
- *Percent Distance Error*: the ratio of the difference between the total distance (TD) and actual distance (AD) over the total distance (TD) as shown below:

$$PDE = \frac{|AD - TD|}{TD} * 100$$

3.2.5 Statistical Analysis

Exploratory data analysis preceded all formal statistical analysis. Descriptive statistics and frequencies were calculated for all measures of interest including the cursor movement measures and the measures collected through the questionnaires.

A mixed model analysis of variance (ANOVA) was used to determine if differences existed between either of the two types of controls (customized switch control versus proportional isometric control) or the two types of diagnosis (athetosis versus dystonia). Cursor

movement measures including the percentage of acquired targets, movement time, reaction time, and percent distance error were analyzed. Statistical analysis was performed using the SPSS statistical package. Alpha level was set at 0.05 for all analysis [44].

3.3 RESULTS

Three women and eight men with an average age of 47.0 ± 10.3 years were enrolled in the study. Six people had athetosis and dystonia, and five people had only dystonia. Ten participants used a power wheelchair as their primary means of mobility and one person used a manual wheelchair and did not self propel.

Nine participants completed the computer access questions outlined in the Customized Control Assistive Device History Survey in Appendix B. Six people reported using a computer at home and four people used a computer outside the home (school or work). Of the six participants who used a computer, none of them used a modified keyboard. Three of the participants used a modified mouse and the other three used a traditional mouse. All computer users used email. Participants rated their internet usage frequency based on task type shown in Figure 12.

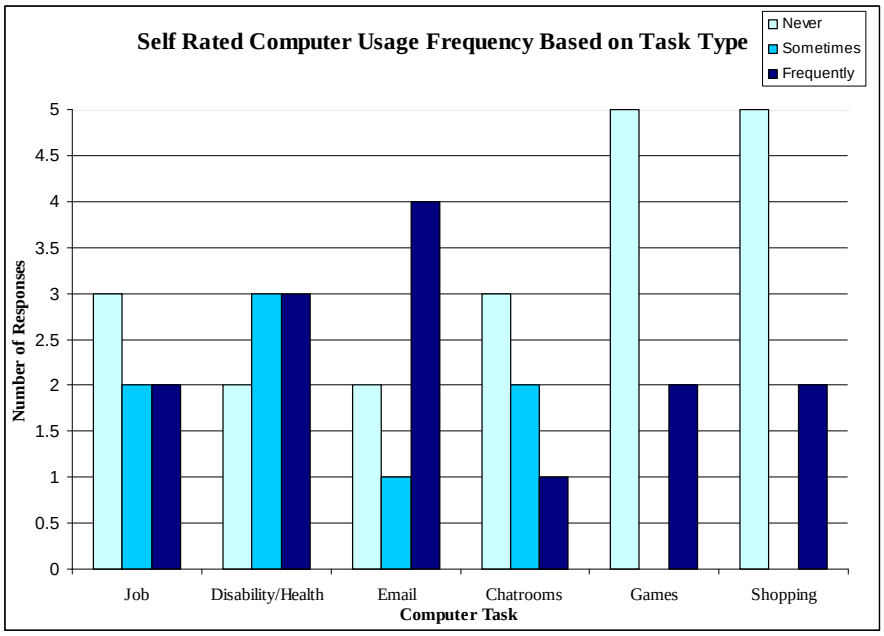

Figure 12. Internet Usage Frequency Based on Task Type

The median of the four Modified Ashworth Scales (MAS) scores for the wrist and elbow joints in flexion and extension in the upper extremity that was used to control the joystick were obtained. Table 5 shows the clinical measures. The MAS and Penn Spasm Frequency Scale scores range from 0 to 5 with higher scores meaning more spasticity. The range of scores for Barthel Index and Global Dystonia Rating Scale is 0-100 with higher scores indicating more independence in activities of daily living and more dystonia. Overall medians were obtained for the participant groups for ordinal scales, and overall means obtained for continuous scales.

	Scores	
Measures	Athetosis (n=4)	Dystonia (n=4)
Median Modified Ashworth Scale (range)	1.25 (0.5-3)	2.5 (2-5)
Median Penn Spasm Frequency Scale (range)	4.0 (1-4)	2.5 (1-4)
Median Strength Scale (range)	3.5 (2-4)	4.0 (1-5)
Mean Global Dystonia Rating Scale (SD)	37.7 (18.4)	18.2 (5.6)
Mean Barthel Index Score (SD)	26.7 (20.6)	23.0 (11.8)

Table 5. Clinical Measurement Scores for Each Movement Disorder Group

Ten participants completed visit 1. Five of them had athetosis and dystonia, and the other five had only dystonia. The number of targets each participant acquired and their disability

group are shown in Table 6. Six participants could not acquire any of the targets and one participant acquired all of the targets.

Participant Number	Movement Disorder Group	% Acquired Targets
1	Dystonia	9.4
2	Athetosis and Dystonia	0
3	Dystonia	100
4	Dystonia	0
5	Athetosis and Dystonia	68.8
6	Athetosis and Dystonia	0
8	Athetosis and Dystonia	6.3
9	Dystonia	0
10	Dystonia	0
11	Athetosis and Dystonia	0

Table 6. Percentage of Acquired Targets during Visit 1

Eight out of the 11 participants completed visit 2. Participants #6 and #9 withdrew from the study after the first visit and participant #7's data was incomplete for both visits. There were an equal number of participants with and without athetosis. Because many of the participants acquired none or very few targets, yet they could independently drive their power wheelchairs, the original switch scanning customization method outlined above was abandoned and alternatively, data was visually inspected in order to prescribe a customized control method.

One of the 3 switch controls (Table 7) were prescribed for the participants as indicated by Table 8. If the participant could move in all four directions (forward, backward, left, right) and sustain cursor movement around targets (even if he/she was unable to acquire a target by hovering within it), he/she was prescribed switch A. Alternatively, if the person could move in all four directions yet could only pass though the targets and not sustain a constant cursor movement around the target, he/she was prescribed switch B. Lastly, if the person did not have directional control, he/she was prescribed switch C which incorporated on-screen scanning. In some cases, the switch chosen for the participant based on the method described above was found to be inappropriate during practice, and a better switch was reassigned as shown in Table 8.

Switch/Scanning	Description
A	Constant force greater than a threshold value in one of the cardinal directions will produce cursor movement in that direction.
B	Quick force greater than a threshold value in one of the cardinal directions will produce movement in that direction until another force exceeding the threshold in any direction is applied to stop the cursor movement.
C	The joystick was used to actively scan (visually shown on computer screen) by exceeding a threshold force in one direction and used to select a movement direction (on the scan interface) by exceeding a threshold force in an opposite direction. A threshold force in any direction was used to stop the cursor motion and continue the on screen scanning.

Table 7. Descriptions of Three Switch Controls

Participant	Movement Disorder Group	Initial Switch	Final Switch
1	Dystonia	A	A
2	Athetosis and Dystonia	C	A
3	Dystonia	A	A
4	Dystonia	A	A
5	Athetosis and Dystonia	B	A
8	Athetosis and Dystonia	B	A
10	Dystonia	B	B
11	Athetosis and Dystonia	B	B

Table 8. Initial and Final Switch Selection for Each Participant

A 2 (Control Type) x 2 (Movement Disorder Group) mixed-model ANOVA revealed that the main effect for the type of control (switch vs. proportional) was significant for percent distance error ($F(1,6) = 361.2$, $p < 0.01$). Percent distance error was significantly higher using proportional control (40.5 ± 26.0 %) than using the customized switch control (53.0 ± 34.4 %). The main effect for the type of movement disorder was not significant for any of the cursor movement measures. Thus, there was no overall difference between the participants with athetosis and dystonia, and those with dystonia only. Table 9 shows the cursor movement measures using customized switch and proportional isometric control and Table 10 shows the cursor movement measures between the two groups.

Measures (n=8)	Mean Value ± Standard Deviation		p-value
	Switch Control	Proportional Control	
Percentage of Acquired Targets (%)	84.0 ± 12.8	81.6 ± 17.8	0.7
Movement Time (seconds)	9.5 ± 2.8	11.0 ± 3.9	0.1
Reaction Time (seconds)	1.1 ± 0.4	1.3 ± 1.6	0.7
Percent Distance Error (%)*	**40.5 ± 26.0**	**53.0 ± 34.4**	**< 0.01**

Table 9. Descriptive Statistics for Cursor Movement Measures Split by Type of Control Interface

Measures	Mean Value ± Standard Deviation		p-value
	Athetosis (n=4)	Dystonia (n=4)	
Percentage of Acquired Targets (%)	82.4 ± 15.4	83.2 ± 15.7	1.0
Movement Time (seconds)	10.5 ± 3.4	10.0 ± 3.6	0.8
Reaction Time (seconds)	0.9 ± 0.5	1.5 ± 1.5	0.4
Percent Distance Error (%)	75.0 ± 11.4	70.0 ± 9.3	0.5

Table 10. Descriptive Statistics for Cursor Movement Measures Split by Type of Movement Disorder

3.4 DISCUSSION

People with athetosis and dystonia have difficulty using the computer which has prompted researchers to explore alternative methods of pointing control. A major goal of this study was to use a quantitative method in order to prescribe switch control. Using a target acquisition task in which the cursor was constantly pulling back to the center point, directional control and the ability to produce sustained forces was evaluated for each participant. The intention of this task was to determine a person's ability to produce graded forces in a specific direction based on the person's ability to acquire eight different targets on the computer screen by hovering over them with the cursor. To the author's knowledge no one has tried to use this specific method to prescribe switch scanning control for people with movement disorders.

Unfortunately, for most participants, this task was foreign due to the cursor returning to center and too difficult for the participants to select the target via hovering over it. Therefore, the switch scanning prescription was done by visually inspecting the plotted data much like a computer access specialist would do in real-time when evaluating a client for an adaptive

computer interface. Although many of these participants were power wheelchair users (though some had much difficulty using proportional control for their own wheelchairs), this method of testing had the advantage of revealing significant motor control deficits in target acquisition, which requires more fine motor control. It was important to use this design to quantify deficits. Using a cursor that did not return to center would have allowed participants to release the control interface and would not have given the researchers the ability to determine whether they could sustain a force in particular direction, or with a certain amplitude. Contrary to what the researchers had anticipated, this was a significant problem for most of the participants.

A significant difference was found in type of control used with respect to percent distance error. Switch control was better for both people with athetosis and dystonia to move on the most direct path to the target. This result is interesting because all of the people who completed visit 2 drove their power wheelchair independently, yet these results indicate that switch control could be advantageous to this population for computer access tasks that involve direct paths.

These results can be deceiving, due to the layout of the targets on the screen. Since the targets were directly above, to the right, below, and to the left of the starting point and the switches were designed to move only in one direction (forward, backward, left and right), percent distance error might have been higher than it would have been if the targets were placed at an angle from the starting point. No significant differences were found between athetosis and dystonia groups (with respect to all cursor movement outcome measures) indicating that a more individualized analysis is necessary in order to determine which control method is appropriate for each end user.

It is also important to note that the group of subjects with athetoid CP were likely more impaired at baseline than those with dystonic CP, due to the differences seen in some of the

clinical measurement scores. However, because there were no differences in performance measures between groups of subjects, the severity of functional impairment did not seem to affect the results.

3.5 CONCLUSIONS

Results of this study indicate that switch / scanning control could be useful to improve cursor trajectory in the most direct path to the target. This type of control could be useful in a scrolling task as well as when navigating through menus when the user needs to move the cursor in a straight line up and down or side to side. However, there is no conclusive evidence that switch / scanning control is the best method of control for this population. Future studies should improve the way data is collected in the first visit in order to customize the switch / scanning control to an individual's needs.

The outcomes of this study might also be useful in the development and prescription of AAC devices and further developing the IJ as an AAC interface for people with CP. The process for prescribing appropriate technology both for computer access and other switch / scanning technology can be long and frustrating both for the end user and the rehabilitation engineer. Instead of using trial and error, research is needed to design more efficient methods to prescribe the appropriate technology to supplement the rehabilitation engineer's professional opinion when prescribing computer access devices and AAC devices.

4.0 FUTURE WORK

Quantification of Cursor Movement in Athetoid and Spastic CP

There is much work to be done in order to improve real time filters for people with athetoid and spastic CP in order to improve computer access. Many other outcome measures could be collected to further pinpoint where and why people have difficulty acquiring targets via a hovering method. Specifically, looking at kinematic measures before and after first target entry would be useful in guiding filter development. Additionally, a submovement analysis based on velocity peaks instead of pauses could lead to additional findings about athetoid and spastic cursor motion. It would also be useful to look at the difference in outcome measures with respect to the distance between the starting point and the target as to determine if people have more trouble with closer targets or more distant targets.

Instantaneous velocity was calculated using the distance and time from one sampling point to another. Alternatively, velocity can be calculated using a spline between several points so the average would be less sensitive to noise. This would act to smooth the data.

Lastly, adding a selection button to the IJ could be beneficial for use by someone who has the ability to click using another part of their body such as their knee or their head. This would enable the person to move to the target using the IJ as a pointer and acquire the target using a switch like the button on a mouse.

Customized Control for People with Athetosis and Dystonia

Little is understood regarding how to quantitatively prescribe switch / scanning control for people with CP. One of the major problems in the customized control study was the data acquisition in the first visit which was supposed to be used in order to customize a control interface for the individual. In order to use the IJ as a switch, the participant needs to be able to

exceed a certain force. If the person can sustain a force in all 4 compass directions, proportional control should be prescribed. However, the user does not need to be able to sustain a force in order to use the joystick as a switch as was assumed by the method of data acquisition in visit 1 of the study. As a minimum they need to be able to consistently exceed a force in one direction.

Therefore, future studies should systematically test both a person's ability to sustain a force in one or more directions and then if they are unable to do so, he or she should be tested to determine if he or she can consistently exceed a force in one or more directions. Once it is determined that the person can either sustain or consistently exceed a force, then further testing should be focused on sustaining or exceeding two distinct forces in a single direction. This method pinpoints a person's specific directional control and graded force abilities as opposed to just relying on their ability to sustain a force and direction at the same time.

This new method for prescribing switch scanning control could be carried out using force gauges which would give the user feedback as to how hard he or she is pushing on a joystick in a specified direction at any given time. Missing targets often caused the participants to become frustrated which negatively affected their performance. Using gauges would eliminate this confounding condition. Additionally, pie shaped areas could be used to replace the targets so the test would rely less on a person's accuracy in a dwelling task and focus more on a person's ability to sustain a force as well as maintain directional control at the same time.

Computer Access and Studies

Future studies researching computer access solutions for people with movement disorders should utilize the methods and outcome measures outlined by the International Standardization Organization in the document "Ergonomic Requirements for Office Work with Visual Display

Terminals (VDTs) – Part 9: Requirements for non-keyboard input devices" [23]. These guidelines would be useful in comparing different populations and interventions.

Using these standards, task precession or the accuracy required in order to complete a pointing task should be considered in the design of the study. ISO defines three precision levels based on the index of difficulty (ID) values. Ideally, there would be an equal number of trials at all three precision levels (ID<=4, 4<ID<=6, ID>6). This value can be increased or decreased based on target width and ideal path distance [23].

The ISO document also outlines a Multi-directional pointing task used to evaluate pointing movements in many different directions (illustrated in Figure 3.B of the ISO document). This task would replicate selecting randomly located icons on the screen, repositioning a cursor at different areas on a screen, and cell selection in a spread sheet. The most important part of this design is that the targets are equidistant apart when comparing one task to another. Task precision can be varied between trials by adjusting the target width or distance between targets.

During the actual test, some people were confused as to which target they were supposed to acquire on the screen. While some people found it intuitive to move from the blue to the yellow target others found it counter intuitive. In the future only one target (the goal target) should appear on the screen at a time. Adding a drop down menu with target color to the screen could also help resolve this problem so the person could choose a color of their choice that could be clearly differentiated from the background.

There is a need for a common questionnaire that could be used for many computer access studies. The advantage of a common questionnaire is that researchers could look at the difference or similarity of responses between different populations. This questionnaire should include all questions that these studies collected and should account for how participants would

like to use the computer in addition to how they actually use it. Knowing how people currently use the computer (if at all) and how they would like to use it would help to show where there are computer access deficiencies. The ISO 9241-9 document also outlines computer access questionnaires that could be useful in evaluating specific computer access technologies.

Lastly, an innovative way to determine where people are having difficulty in completing computer tasks is to sample their cursor movements using their traditional pointing device in their natural environment as opposed to a testing environment. This could be done by monitoring pointing activity via a website set up to record cursor movements. It could also be used to compare people's performance with different pointing devices as well as the time it takes to fully adapt to a new device.

APPENDIX A

Medical Information Forms

A.1 THE MODIFIED ASHWORTH SCALE

The Modified Ashworth Scale

Score	
0	No increase in muscle tone
1	Slight increase in muscle tone, manifested by a catch and release or by minimal resistance at the end of the range of motion when the affected part(s) is moved in flexion or extension.
2	Slight increase in muscle tone, manifested by a catch, followed by minimal resistance throughout the reminder (less than half) of the ROM (range of movement).
3	More marked increase in muscle tone through most of the ROM, but affected part(s) easily moved.
4	Considerable increase in muscle tone passive, movement difficult.
5	Affected part(s) rigid in flexion or extension.

	Left	**Right**
Wrist Flexors:	_____	_____
Wrist Extensors:	_____	_____
Elbow Flexors:	_____	_____
Elbow Extensors:	_____	_____

A.2 THE BARTHEL INDEX

THE BARTHEL INDEX

Patient Name: _____
Rater Name: _____
Date: _____

Activity	Score
FEEDING 0 = unable 5 = needs help cutting, spreading butter, etc., or requires modified diet 10 = independent	_____
BATHING 0 = dependent 5 = independent (or in shower)	_____
GROOMING 0 = needs to help with personal care 5 = independent face/hair/teeth/shaving (implements provided)	_____
DRESSING 0 = dependent 5 = needs help but can do about half unaided 10 = independent (including buttons, zips, laces, etc.)	_____
BOWELS 0 = incontinent (or needs to be given enemas) 5 = occasional accident 10 = continent	_____
BLADDER 0 = incontinent, or catheterized and unable to manage alone 5 = occasional accident 10 = continent	_____
TOILET USE 0 = dependent 5 = needs some help, but can do something alone 10 = independent (on and off, dressing, wiping)	_____
TRANSFERS (BED TO CHAIR AND BACK) 0 = unable, no sitting balance 5 = major help (one or two people, physical), can sit 10 = minor help (verbal or physical) 15 = independent	_____
MOBILITY (ON LEVEL SURFACES) 0 = immobile or < 50 yards 5 = wheelchair independent, including corners, > 50 yards 10 = walks with help of one person (verbal or physical) > 50 yards 15 = independent (but may use any aid; for example, stick) > 50 yards	_____
STAIRS 0 = unable 5 = needs help (verbal, physical, carrying aid) 10 = independent	_____
TOTAL (0–100):	_____

A.3 GOLBAL DYSTONIA RATING SCALE

Global Dystonia Rating Scale

Rater Name:_____
Participant ID:_____
Date:_____

BODY AREA **SCORE**

Rating Scale
0 = No dystonia present in that body area
1 = Minimal dystonia
5 = Moderate dystonia
10 = Most severe dystonia

EYES AND UPPER FACE _____

LOWER FACE _____

JAW AND TONGUE _____

LARYNX _____

NECK _____

SHOULDER AND PROXIMAL ARM _____

DISTAL ARM AND HAND INCLUDING ELBOW _____

PELVIS AND UPPER LEG _____

DISTAL LEG AND FOOT _____

TRUNK _____

TOTAL _____

A.4 BARRY-ALBRIGHT DYSTONIA SCALE

Barry-Albright Dystonia Scale

Participant ID: _____
Rater Name: _____
Date: _____

* = unable to assess

Body Part **Score**

EYES: signs of dystonia of the eyes include: prolonged eyelid spasms, and/or forced eye deviations.

0 = absence of eye dystonia _____
1= Slight. Dystonia less than 10% of the time and does not interfere with tracking.
2 = Mild. Frequent blinking without prolonged spasms of eye closure, and/or eye movements less than 50% of the time.
3 = Moderate. Prolonged spasms of eyelid closure, but eyes open most of the time, and/or eye movements more than 50% of the time that interfere with tracking, but able to resume tracking.
4 = Severe. Prolonged spasms of eyelid closure, with eyes closed at least 30% of the time, and/or eye movements more than 50% of the time that prevent tracking.

MOUTH: signs of dystonia of the mouth include: grimacing, clenched or deviated jaw, forced open mouth, and/or forceful tongue thrusting.

0 = Absence of mouth dystonia. _____
1 = Slight. Dystonia less than 10% of the time and does not interfere with speech and/or feeding.
2 = Mild. Dystonia less than 50% of the time and does not interfere with speech and/or feeding.
3 = Moderate. Dystonia more than 50% of the time, and/or dystonia that interferes with speech and/or feeding.
4 = Severe. Dystonia more than 50% of the time, and/ or dystonia that prevents speech and/or feeding.

NECK: signs of dystonia of the neck include: pulling of the neck into Any plane of motion: extension, flexion, lateral flexion or rotation.

0 = Absence of neck dystonia. _____
1 = Slight. Pulling less than 10% of the time and does not interfere with lying, sitting, standing and/or walking.
2 = Mild. Pulling less than 50% of the time and does not interfere with lying, sitting, standing and/or walking.
3 = Moderate. Pulling more than 50% of the time and/or dystonia that interferes with lying, sitting, standing and/or walking.
4 = Severe. Pulling more than 50% of the time and/ or dystonia that prevents sitting in standard wheelchair, standing and/or walking.

TRUNK: signs of dystonia of the trunk include: pulling of the trunk into Any plane of motion: extension, flexion, lateral flexion or rotation. _____

0 = Absence of trunk dystonia.
1 = Slight. Pulling less than 10% of the time and does not interfere with lying, sitting, standing and/or walking.
2 = Mild. Pulling less than 50% of the time and does not interfere with lying, sitting, standing and/or walking.
3 = Moderate. Pulling more than 50% of the time, and/or dystonia that interferes with lying, sitting, standing and/or walking.
4 = Severe. Pulling more than 50% of the time and/or dystonia that prevents positioning in standard wheelchair, standing and/or walking (e.g. requires adapted seating system to control posture, such as ASIS bar)

UPPER EXTREMITIES: signs of dystonia of the upper extremities include: sustained muscle contractions causing abnormal posturing of the upper extremities. **Left:** _____ Right**:** _____

0 = Absence of upper extremity dystonia.
1 = Slight. Dystonia less than 10% of the time and does not interfere with normal positioning and/or functional activities.
2 = Mild. Dystonia less than 50% of the time and does not interfere with normal positioning and/or functional activities.
3 = Moderate. Dystonia more than 50% of the time and/or dystonia that interferes with normal positioning and/or upper extremity function.
4 = Severe. Dystonia more than 50% of the time and/or dystonia that interferes with normal positioning and/or upper extremity function. (e.g. arms restrained in wheelchair to prevent injury)

LOWER EXTREMITIES: signs of dystonia of the lower extremities include: sustained muscle contractions causing abnormal posturing of the lower extremities. **Left:** _____ **Right:** _____

0 = Absence of lower extremity dystonia.
1 = Slight. Dystonia less than 10% of the time and does not interfere with normal positioning and/or functional activities.
2 = Mild. Dystonia less than 50% of the time and does not interfere with normal positioning and/or functional activities.
3 = Moderate dystonia more than 50% of the time and/or dystonia that interferes with normal positioning and/or lower extremity weight bearing or function.
4 = Severe. Dystonia more than 50% of the time and/or dystonia that prevents normal positioning and/or lower extremity weight bearing and/or function. (e.g. cannot maintain standing due to severe dystonia at ankles).

DIRECTIONS: Assess the patient for dystonia in each of the following regions: eyes, mouth, neck, trunk, and each upper and lower extremity (8 body regions). Write the scores on the lines provided. Rate severity based only on dystonia as evidenced by abnormal movements or postures. When assessing functional limitations, do not score dystonia based on other factors, such as weakness, lack of motor control, cognitive deficits, primitive reflexes, and/or other movement disorders as defined below.

Dystonia: Sustained muscle contractions causing twisting and repetitive movements or abnormal postures.

Spasticity: Velocity-dependent resistance to passive stretch.

Athetosis: Distal writhing or contorting movements.

Chorea: Brief, rapid, unsustained, irregular movements.

Ataxia: Incoordination of movement characterized by wide based unsteady gait, flailing movements.

APPENDIX B

Questionnaires

B.1 NONLINEAR FILTERING OF ATHETOID MOVEMENT GENERAL INFORMATION AND COMPUTER USAGE QUESTIONAIRE

Nonlinear Filtering of Athetoid Motion

University of Pittsburgh IRB #: 0605040

Date: ____/____/____

Date of Birth: ____/____/____

Gender: ___Male ___Female

Veteran: ___Yes ___No

Ethnic Origin:
___African American/Black
___Asian American
___Caucasian/White
___Hispanic
___Other: _____

Are you currently using:
___ Manual Wheelchair
___ Power Wheelchair
___ Scooter
___ Other (please specify): _____
___ No Wheeled Mobility Device

Wheelchair/Scooter Make (brand):

☐ Action/Invacare ☐ Permobil
☐ Everest and Jennings ☐ Pride
☐ Kuschall ☐ Sunrise/Quickie
☐ Otto Bock ☐ TiLite/TiSport
☐ Other (please specify): _____

Wheelchair/Scooter Model: _____

What date did you start using a wheelchair/scooter: ____/____/____
If you have a power wheelchair/scooter, please indicate your ability level with the device:

- ☐ I cannot drive at all; I need an attendant to move me
- ☐ I can drive a little but only indoors with few obstacles
- ☐ I can drive indoors and/or outdoors with some assistance
- ☐ I can drive indoors and/or outdoors without any assistance

Please describe your fatigue condition during your daily activities:

- ☐ Constantly
- ☐ Off and on
- ☐ Rarely

Please describe your pain condition during your daily activities:

- ☐ Constantly
- ☐ Off and on
- ☐ Rarely

Are you left or right handed?

- ☐ Right handed
- ☐ Left handed

Can you independently use the computer?

- ☐ Yes
- ☐ No

If yes, what is your control interface?

- ☐ Regular hand-operated mouse
- ☐ Regular hand-operated keyboard
- ☐ Regular hand-operated mouse and keyboard
- ☐ Other. Please specify _____

How often do you use a computer?

- ☐ Never
- ☐ Several times a week
- ☐ Several times a month
- ☐ Other, Please specify _____

Please describe your spasm frequency (based on Penn Spasm Frequency scale)

- ☐ No spasms
- ☐ Spasms that occur only if stimulated
- ☐ Less than one spasm per hour, occur often without stimulation
- ☐ More than one spasm per hour, but less than 10 per hour, occur often without stimulation
- ☐ More than 10 spasms per hour, occur often without stimulation

B.2 CUSTOMIZED CONTROL GENERAL INFORMATION

Customized Control to Enable Function in Dystonia and Choreoathetosis

University of Pittsburgh IRB #: 0611035

Date: ____/____/____ **Time: _____ AM/PM (circle)**

Date of Birth: ____/____/____ **Age:_____**

Gender: ☐ Male ☐ Female

Veteran: ☐ Yes ☐ No

Ethnic Origin:
☐ African American/Black
☐ Asian American
☐ Caucasian/White
☐ Hispanic
☐ Other: _____

Injury or Diagnosis: _____

Date of Injury or date of diagnosis: ____/____/____

What are you currently using as your primary means of mobility?
☐ Manual Wheelchair
☐ Power Wheelchair
☐ Pushrim Activated Power Assist System
☐ Scooter

Make: _____ **Model: _____**

B.3 CUSTOMIZED CONTROL ASSISITVE DEVICE HISTORY QUESTIONAIRE

ASSISTIVE DEVICE HISTORY

Can you do any standing?	Yes	No
Can you do any walking?	Yes	No
If you can walk, are you able to walk		
For 150 feet in your home?	Yes	No
For one street block outside?	Yes	No
Up one flight of stairs?	Yes	No

Which aids do you use?

Straight Cane	Yes	No		
Quad Cane	Yes	No		
Walker	Yes	No		
Forearm Crutches (Loftstrand)	Yes	No		
Arm Crutches (Axillary)	Yes	No		
AFO (ankle foot orthosis, short leg brace)	Yes	No	Right	Left
KAFO (knee ankle foot orthosis, long leg brace)	Yes	No	Right	Left
Are your braces	Plastic	Metal		
Have you ever had skin breakdown from your braces?	Yes	No		
Does swelling interfere with how the braces fit?	Yes	No		

Other_____

Which devices do you use?

Manual wheelchair	Yes	No
Scooter	Yes	No
Electric Power wheelchair	Yes	No
If yes, do you have:		
Tilt	Yes	No
Recline	Yes	No
Elevating Leg Rests	Yes	No
Standing	Yes	No
Seat Elevator	Yes	No
Power Assist wheelchair	Yes	No

Wheelchair types

Manufacturer _____ model _____

_____ _____

Do you use a wheelchair or scooter over 40 hours per week?	Yes	No	
If you have two devices, which do you use most often?	Manual	or	Power
Funding source for the most often used wheelchair	_____		
How many times in the past 6 mo. has your most often used device been repaired?	_____		
Has any of the following occurred because of <u>breakdown</u> of your device			
Stranded at or away from home	Yes	No	
Injury	Yes	No	
Missed work or school	Yes	No	
Missed medical appointment	Yes	No	

Other Devices

Do you use a computer at home	Yes	No
Do you use a computer outside the home	Yes	No
Do you use any of the following computer devices		
Voice Activation	Yes	No
Voice Recognition	Yes	No
Mouth Stick	Yes	No
Head Pointer	Yes	No
Foot pedals	Yes	No
Eye control	Yes	No
Typing brace or splint for hand	Yes	No
Modified or on screen keyboard	Yes	No
Modified mouse	Yes	No
Other _____		

How often do you use email?	Never
	Daily
	Weekly
	Monthly
If you use email, do you use it at	Home
	Work or School
	Library, café, wirelesses, etc.

How often do you use the internet for

Job information	Never	Sometimes	Frequently
Disability or health information	Never	Sometimes	Frequently
Email	Never	Sometimes	Frequently
Chatrooms	Never	Sometimes	Frequently
Games	Never	Sometimes	Frequently
Shopping	Never	Sometimes	Frequently
Other_____		Sometimes	Frequently

Do you or a family member own a modified		
Car or SUV	Yes	No
Van	Yes	No
Truck	Yes	No

Do you drive any of these vehicles	Yes	No
Do you own a cell/mobile phone	Yes	No

List any sports that you play and any adaptive sports devices that you use:

Describe any injuries that you have sustained and what they were from (using a wheelchair or device, during sports, etc):

List any other devices that you use that you feel are important to you:

APPENDIX C

C++ Builder Graphical User Interface Code

C.1 NONLINEAR FILTERING DATA COLLECTION CODE

```
//---------------------------------------------------------------------------
//Nonlinear Filtering Visit 1 - Screening
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Unit1.h"
#include "Unit2.h"
#define PI 3.1415926
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CGAUGES"
#pragma resource "*.dfm"
#pragma resource "extrares.RES"
TForm2 *Form2;
//---------------------------------------------------------------------------
//radius of 96/2 = 48 = 1 inch (96 pixel) diameter target
int radius = 48;
AnsiString subject_id;
HANDLE hComm=NULL;
COMMTIMEOUTS ctmoNew={0},ctmoOld;
DWORD dwBytesRead;
DWORD dwEvent,dwError;
COMSTAT cs;
struct time bt, et;
int count1;
int direction1, speed1;
int data1[40000];
int scale=4;
//initial position of the axquired target
int xtarget=350, ytarget=350;
//target position of the new goal target

//cursor position
int xplot=350, yplot=350, hit =1 , miss=0;
int flag=0, flag1=0, count=0, trialend=0, trialtime=0, first=0, aflag=0;

int canvasflag =1;
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void canvas_draw()
{
      if (canvasflag==1)
            {
             direction1=2048;
             speed1=2048;
             canvasflag=0;
            }

       if (direction1<=2015+20 & direction1>=2015-20)direction1=2048;
       if (speed1<=2040+20 & speed1>=2040-20) speed1=2048;
       if (xplot >= 800) xplot = 800;
       if (xplot <= 3) xplot = 3;
       if (yplot >= 680) yplot = 680;
       if (yplot <= 3) yplot = 3;
       xplot= (xplot + (10*(direction1-2048)/2048));
       yplot= yplot-(10*(speed1-2048)/2048);
       Form2->Canvas->Brush->Color = clAqua;
       Form2->Canvas->Pen->Color = clAqua;
       Form2->Canvas->Pen->Width = 5;
       Form2->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
}
void draw_target(TColor cld, int x, int y)
{
       Form2->Canvas->Brush->Color = cld;
       Form2->Canvas->Pen->Color = clBlack;
       Form2->Canvas->Pen->Width = 3;
       Form2->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
       Form2->Canvas->Brush->Color = clBlack;
       Form2->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
       Form2->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
       Form2->Canvas->Brush->Color = cld;
       Form2->Canvas->Pen->Color = clBlack;
       Form2->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
       Form2->Canvas->Brush->Color = cld;
       Form2->Canvas->Pen->Color = cld;
       Form2->Canvas->Pen->Width = 3;
       Form2->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
 }
void draw_target_smile(TColor cld, int x, int y)
{
       Form2->Canvas->Brush->Color = cld;
       Form2->Canvas->Pen->Color = clBlack;
       Form2->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
       Form2->Canvas->Brush->Color = clBlack;
       Form2->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
       Form2->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
       Form2->Canvas->Brush->Color = clWhite;
       Form2->Canvas->Pen->Color = clBlack;
       Form2->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
       Form2->Canvas->Brush->Color = cld;
       Form2->Canvas->Pen->Color = cld;
       Form2->Canvas->Pen->Width = 3;
       Form2->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
}
void draw_target_frown(TColor cld, int x, int y)
```

```
{
	Form2->Canvas->Brush->Color = cld;
	Form2->Canvas->Pen->Color = clBlack;
	Form2->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
	Form2->Canvas->Brush->Color = clBlack;
	Form2->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
	Form2->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
	Form2->Canvas->Brush->Color = cld;
	Form2->Canvas->Pen->Color = clBlack;
	Form2->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.1),x+(radius*0.6),y+(radius*0.7));
	Form2->Canvas->Brush->Color = cld;
	Form2->Canvas->Pen->Color = cld;
	Form2->Canvas->Pen->Width = 3;
	Form2->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.4),x+(radius*0.6),y+(radius*0.75));
}
int inside()
{
	int timein;
	if (flag==1)
	{
	trialtime++;
		if ((xplot-xtarget)*(xplot-xtarget)+(yplot-ytarget)*(yplot-ytarget)<radius*radius)
		{
		//inside the target
		flag1=1;
		count++;
		timein=10*count;
		Form2->Canvas->Brush->Color = clRed;
		Form2->Canvas->Pen->Color = clRed;
		Form2->Canvas->Pen->Width = 5;
		Form2->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
		}
		else
		{
		//outside the target
		flag1=0;
		count=0;
		Form2->Canvas->Brush->Color = clBlue;
		Form2->Canvas->Pen->Color = clBlue;
		Form2->Canvas->Pen->Width = 5;
		Form2->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
		}
	if (flag1==1 & timein==2000)
		{
		draw_target(clRed, xtarget, ytarget);
		trialend=1;
		trialtime=0;
		first=0;
		return 1;
		}
	else if (flag1==0 & trialtime*10>=20000)
		{
		draw_target(clYellow, xtarget, ytarget);
		trialend=2;
		trialtime=0;
		first=0;
```

```
            return 1;
            }
        else return 0;
        }
        else return 0;
}

void generate_target()
{
Form2->Refresh();
xtarget=350;
ytarget=200-(radius);
xplot = 350;
yplot = 350;
draw_target(clYellow, xtarget, ytarget);
}
//---------------------------------------------------------------------------
//Click Button 1
void __fastcall TForm2::Button1Click(TObject *Sender)
{
        Form2->Refresh();
        generate_target();
        xplot=350;
        yplot=350;

        Timer1->Enabled=true;
        Button1->Enabled=false;
        flag=1;

        unsigned int count1=0;
        int ab11, ab21;

// Serial Port Communication
        DCB dcbCommPort;
        hComm=CreateFile("COM1",GENERIC_READ|GENERIC_WRITE,0,
            NULL,OPEN_EXISTING,0,NULL);
        GetCommState(hComm,&dcbCommPort);
        BuildCommDCB("38400,N,8,1", &dcbCommPort);
        SetCommState(hComm,&dcbCommPort);
        SetCommMask(hComm,EV_RXCHAR);
        GetCommTimeouts(hComm,&ctmoNew);
        ctmoNew.ReadIntervalTimeout=2.7;
        ctmoNew.ReadTotalTimeoutConstant=0;
        ctmoNew.ReadTotalTimeoutMultiplier=0;
        SetCommTimeouts(hComm,&ctmoNew);

        // Get start time
        gettime(&bt);
        while(1)
        {
            ClearCommError(hComm,&dwError,&cs);
            unsigned char InBuff[10];
            DWORD dwBytesRead;
            Application->ProcessMessages();
            while(1)
            {
```

```
                if (WaitCommEvent(hComm,&dwEvent,NULL))
                    if (dwEvent & EV_RXCHAR)
                    {
                        ClearCommError(hComm,&dwError,&cs);
                        ReadFile(hComm,InBuff,4,&dwBytesRead,NULL);
                        ClearCommError(hComm,&dwError,&cs);
                        ab11=(unsigned int)(unsigned char)(InBuff[0]);
                        if ((ab11&240)!=16) break;
                        ab21=(unsigned int)(unsigned char)(InBuff[2]);
                        if ((ab21&240)!=32) break;
                        direction1=data1[count1*2]=(ab11&15)*256+(unsignedint)(unsigned
                          char)(InBuff[1]);
                        speed1=data1[count1*2+1]=(ab21&15)*256+(unsignedint)(unsigned
                          char)(InBuff[3]);
                        ++count1;
                        break;
                    }
            }
            gettime(&et);
            if (hit+miss >= 11) {break;}
        }
        Timer1->Enabled=false;
        SetCommMask(hComm,0);
        PurgeComm(hComm,PURGE_RXABORT);
        SetCommTimeouts(hComm,&ctmoOld);
        CloseHandle(hComm);
        Button1->Enabled=true;
}
//---------------------------------------------------------------------------
//Timer
void __fastcall TForm2::Timer1Tick(TObject *Sender)
{
    char strBuffer[9];
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", bt.ti_hour, bt.ti_min, bt.ti_sec, bt.ti_hund);
    Edit3->Text=strBuffer;
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", et.ti_hour, et.ti_min, et.ti_sec, et.ti_hund);
    Edit4->Text=strBuffer;
    Edit1->Text=IntToStr(direction1);
    Edit2->Text=IntToStr(speed1);
    Edit5->Text=IntToStr(hit-1);
    Edit6->Text=IntToStr(miss);
    if (flag==1 & !inside() & !trialend ) canvas_draw();
    if (trialend==1 & first==0)
    {
      Timer1->Interval=1000;
      draw_target_smile(clRed,xtarget,ytarget);
      aflag=1;
      first=1;
      hit=hit+1;
      Edit5->Text=IntToStr(hit-1);
    }
    else if (trialend==2 & first==0)
    {
      Timer1->Interval=1000;
      draw_target_frown(clLime,xtarget,ytarget);
```

```
      aflag=1;
      first=1;
      miss=miss+1;
      Edit6->Text=IntToStr(miss);
    }
    if (trialend!=0 & aflag==0)
    {
      generate_target();
      Timer1->Interval=10;
      trialend=0;
    }
    if (first==1) aflag=0;
    /*if (flag==1 & !inside() & !trialend ) canvas_draw();
    if (trialend==1 & first==0)
    {
      Timer1->Interval=1000;
      draw_target(clRed,xtarget,ytarget);
      aflag=1;
      first=1;
      hit=hit+1;
    }
    else if (trialend==2 & first==0)
    {
      Timer1->Interval=1000;
      draw_target(clYellow,xtarget,ytarget);
      aflag=1;
      first=1;
      miss=miss+1;
    }
    if (trialend!=0 & aflag==0)
    {
      generate_target();
      Timer1->Interval=10;
      trialend=0;
    }
    if (first==1) aflag=0;*/
}
void __fastcall TForm2::Form2Paint(TObject *Sender)
{                                       Canvas->Brush->Color = clBlack;
      Canvas->Pen->Color = clBlack;
      Canvas->Ellipse(350-7,350-7,350+7,350+7);
}
void __fastcall TForm2::TemplateBias1Click(TObject *Sender)
{
      Form2->Visible=false;
      Form1->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
      Form1->Visible=true;
      Form2->Visible=false;
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
//Nonlinear Filtering Visit 1 – Target Acquisition Task
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Unit1.h"
#include "Unit2.h"
#define PI 3.1415926
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CGAUGES"
#pragma resource "*.dfm"
#pragma resource "extrares.RES"
TForm1 *Form1;
//---------------------------------------------------------------------------
//radius of 96/4 = 24 = 1/2 inch (48 pixels) in diameter target
int radius = 48;
extern AnsiString subject_id;
HANDLE hComm=NULL;
COMMTIMEOUTS ctmoNew={0},ctmoOld;
DWORD dwBytesRead;
DWORD dwEvent,dwError;
COMSTAT cs;
struct time bt, et;
extern int count1;
extern int direction1, speed1;
int data[400000];
extern int scale=4, num = 600 + radius;
//initial position of the axquired target
extern int xtarget=350, ytarget=350;
//target position of the new goal target
extern int pxtarget=350, pytarget=350;
//cursor position
extern int xplot=350, yplot=350, hit=1, miss=0, joystickdata;
extern int flag=0, flag1=0, count=0, trialend=0, trialtime=0, first=0, aflag=0;
int trials = 0;
extern int canvasflag=1;
FILE *fp;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void canvas_draw_dynamic()
{
      if (canvasflag==1)
            {
              direction1=2048;
              speed1=2048;
              canvasflag=0;
            }
```

```
        if (direction1<=2015+20 & direction1>=2015-20)direction1=2048;
        if (speed1<=2040+20 & speed1>=2040-20) speed1=2048;
        if (xplot >= 800) xplot = 800;
        if (xplot <= 3) xplot = 3;
        if (yplot >= 680) yplot = 680;
        if (yplot <= 3) yplot = 3;
        xplot= (xplot + (10*(direction1-2048)/2048));
        yplot= yplot-(10*(speed1-2048)/2048);
        Form1->Canvas->Brush->Color = clAqua;
        Form1->Canvas->Pen->Color = clAqua;
        Form1->Canvas->Pen->Width = 5;
        Form1->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
        fprintf(fp, "%d %d %d %d %d %d %d %d %d %d %d %d\n", direction1, speed1, xplot, yplot, xtarget,
        ytarget, pxtarget, pytarget, miss+(hit-1), miss, hit-1, trials);
}
void draw_target_dynamic(TColor cld, int x, int y)
{
        Form1->Canvas->Brush->Color = cld;
        Form1->Canvas->Pen->Color = clBlack;
        Form1->Canvas->Pen->Width = 3;
        Form1->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form1->Canvas->Brush->Color = clBlack;
        Form1->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form1->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form1->Canvas->Brush->Color = cld;
        Form1->Canvas->Pen->Color = clBlack;
        Form1->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
        Form1->Canvas->Brush->Color = cld;
        Form1->Canvas->Pen->Color = cld;
        Form1->Canvas->Pen->Width = 3;
        Form1->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
        }
void draw_target_smile_dynamic(TColor cld, int x, int y)
{
        Form1->Canvas->Brush->Color = cld;
        Form1->Canvas->Pen->Color = clBlack;
        Form1->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form1->Canvas->Brush->Color = clBlack;
        Form1->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form1->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form1->Canvas->Brush->Color = clWhite;
        Form1->Canvas->Pen->Color = clBlack;
        Form1->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
        Form1->Canvas->Brush->Color = cld;
        Form1->Canvas->Pen->Color = cld;
        Form1->Canvas->Pen->Width = 3;
        Form1->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
}
void draw_target_dynamic_frown(TColor cld, int x, int y)
{
        Form1->Canvas->Brush->Color = cld;
        Form1->Canvas->Pen->Color = clBlack;
        Form1->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form1->Canvas->Brush->Color = clBlack;
        Form1->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
```

```
	Form1->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
	Form1->Canvas->Brush->Color = cld;
	Form1->Canvas->Pen->Color = clBlack;
	Form1->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.1),x+(radius*0.6),y+(radius*0.7));
	Form1->Canvas->Brush->Color = cld;
	Form1->Canvas->Pen->Color = cld;
	Form1->Canvas->Pen->Width = 3;
	Form1->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.4),x+(radius*0.6),y+(radius*0.7));
}
int insider()
{
	int timein;
	if (flag==1)
	{
	trialtime++;
		if ((xplot-xtarget)*(xplot-xtarget)+(yplot-ytarget)*(yplot-ytarget)<radius*radius)
		{//inside the target
		flag1=1;
		count++;
		timein=10*count;
		Form1->Canvas->Brush->Color = clRed;
		Form1->Canvas->Pen->Color = clRed;
		Form1->Canvas->Pen->Width = 5;
		Form1->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
		}
		else
		{
		//outside the target
		flag1=0;
		count=0;
		Form1->Canvas->Brush->Color = clBlue;
		Form1->Canvas->Pen->Color = clBlue;
		Form1->Canvas->Pen->Width = 5;
		Form1->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
		}
	if (flag1==1 & timein==2000)
		{
		draw_target_dynamic(clRed, xtarget, ytarget);
		trialend=1;
		trialtime=0;
		first=0;
		return 1;
		}
	else if (flag1==0 & trialtime*10>=20000)
		{
		draw_target_dynamic(clYellow, xtarget, ytarget);
		trialend=2;
		trialtime=0;
		first=0;
		return 1;
		}
	else return 0;
	}
	else return 0;
}
void generate_targets()
```

```
{
randomize();
Form1->Refresh();
pxtarget=xtarget;
pytarget=ytarget;
xtarget=random(num)+(radius);
ytarget=random(num)+(radius);
//Make sure that targets do not over lap
if ((xtarget > (pxtarget + radius) || xtarget < (pxtarget + radius)) && (ytarget > (pytarget + radius) || ytarget
 < (pytarget + radius)))
{
draw_target_dynamic(clYellow, xtarget, ytarget);
}
draw_target_dynamic(clBlue, pxtarget, pytarget);
xplot = pxtarget;
yplot = pytarget;
}
//---------------------------------------------------------------------------
//Click Button 1
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        Form1->Refresh();
        generate_targets();
        xplot=pxtarget;
        yplot=pytarget;
        trials = trials + 1;
        fp= fopen("C:\Joystick_data_visit_1_25.txt","a");
        miss = 0;
        hit = 1;
        Timer1->Enabled=true;
        Button1->Enabled=false;
        flag=1;
        unsigned int count1=0;
        int ab11, ab21;
// Serial Port Communication
        DCB dcbCommPort;
         hComm=CreateFile("COM1",GENERIC_READ|GENERIC_WRITE,0,
        NULL,OPEN_EXISTING,0,NULL);
        GetCommState(hComm,&dcbCommPort);
        BuildCommDCB("38400,N,8,1", &dcbCommPort);
        SetCommState(hComm,&dcbCommPort);
        SetCommMask(hComm,EV_RXCHAR);
        GetCommTimeouts(hComm,&ctmoNew);
        ctmoNew.ReadIntervalTimeout=2.7;
        ctmoNew.ReadTotalTimeoutConstant=0;
        ctmoNew.ReadTotalTimeoutMultiplier=0;
        SetCommTimeouts(hComm,&ctmoNew);
// Get start time
        gettime(&bt);
        while(1){
                ClearCommError(hComm,&dwError,&cs);
                unsigned char InBuff[10];
                DWORD dwBytesRead;
                Application->ProcessMessages();
                while(1)
                {
```

```
                if (WaitCommEvent(hComm,&dwEvent,NULL))
                    if (dwEvent & EV_RXCHAR)
                    {
                        ClearCommError(hComm,&dwError,&cs);
                        ReadFile(hComm,InBuff,4,&dwBytesRead,NULL);
                        ClearCommError(hComm,&dwError,&cs);
                        ab11=(unsigned int)(unsigned char)(InBuff[0]);
                        if ((ab11&240)!=16) break;
                        ab21=(unsigned int)(unsigned char)(InBuff[2]);
                        if ((ab21&240)!=32) break;
                        direction1=data[count1*2]=(ab11&15)*256+(unsigned int)(unsigned
                          char)(InBuff[1]);
                        speed1=data[count1*2+1]=(ab21&15)*256+(unsigned int)(unsigned
                          char)(InBuff[3]);
                        ++count1;
                        break;
                     }
            }
            gettime(&et);
            if ((miss+hit)>=11)
            {
            fprintf(fp, "%d %d %d %d %d %d %d %d %d %d %d %d\n", direction1, speed1, xplot, yplot,
                    xtarget, ytarget, pxtarget, pytarget, miss+(hit-1), miss, hit-1, trials);
            break;
            }
        }
        Timer1->Enabled=false;
        SetCommMask(hComm,0);
        PurgeComm(hComm,PURGE_RXABORT);
        SetCommTimeouts(hComm,&ctmoOld);
        CloseHandle(hComm);
        Button1->Enabled=true;
}
//---------------------------------------------------------------------------
//Timer
void __fastcall TForm1::Timer1Tick(TObject *Sender)
{
    char strBuffer[9];
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", bt.ti_hour, bt.ti_min, bt.ti_sec, bt.ti_hund);
    Edit3->Text=strBuffer;
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", et.ti_hour, et.ti_min, et.ti_sec, et.ti_hund);
    Edit4->Text=strBuffer;
    Edit1->Text=IntToStr(direction1);
    Edit2->Text=IntToStr(speed1);
    Edit5->Text=IntToStr(hit-1);
    Edit6->Text=IntToStr(miss);
    Edit7->Text=IntToStr(trials);
    if (flag==1 & !insider() & !trialend )
    canvas_draw_dynamic();
    if (trialend==1 & first==0)
    {
      Timer1->Interval=1000;
      draw_target_dynamic(clRed,xtarget,ytarget);
      aflag=1;
      first=1;
      hit=hit+1;
```

```
      Edit5->Text=IntToStr(hit-1);
    }
    else if (trialend==2 & first==0)
    {
      Timer1->Interval=1000;
      draw_target_dynamic_frown(clLime,xtarget,ytarget);
      aflag=1;
      first=1;
      miss=miss+1;
      Edit6->Text=IntToStr(miss);
    }
    if (trialend!=0 & aflag==0)
    {
      generate_targets();
      Timer1->Interval=10;
      trialend=0;
    }
    if (first==1) aflag=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
      fclose(fp);
      Form1->Close();
      Form2->Close();
}
//---------------------------------------------------------------------------
```

C.2 CUSTOMIZED CONTROL DATA COLLECTION CODE

```
//---------------------------------------------------------------------------
//Customixed Control Visit 1
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#define PI 3.1415926
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CGAUGES"
#pragma resource "*.dfm"
TPoint Holes[8] = {TPoint(350, 200),TPoint(500, 350),TPoint(350, 500),
                TPoint(200, 350),TPoint(350, 50),TPoint(650, 350),
                TPoint(350, 650),TPoint(50, 350)};
TForm4 *Form4;
FILE*settings;
FILE*fp;
//---------------------------------------------------------------------------
//radius of 96/2 = 48 = 1 inch (96 pixel) diameter target
int radius = 48;
AnsiString subject_id;
HANDLE hComm=NULL;
COMMTIMEOUTS ctmoNew={0},ctmoOld;
DWORD dwBytesRead;
DWORD dwEvent,dwError;
COMSTAT cs;
struct time bt, et;
int count1;
int direction1, speed1, dir, spd;
int data1[400000];
int scale=4;
int Stamp = 0;
//initial position of the axquired target
int xtarget, ytarget;
//cursor position
int xplot=350, yplot=350, hit =1 , miss=0, t;
int flag=0, flag1=0, count=0, trialend=0, trialtime=0, first=0, aflag=0;
int p0=0, p1=0, p2=0, p3=0, p4=0, p5=0, p6=0, p7=0;
int canvasflag =1;
int Subject_ID, Dead_Zone_Shape_Status, Dead_Zone_Rad_X, Dead_Zone_Rad_Y,
Bias_Axis_Enabled_Status, Bias_Axis_Angle, Gain_Status, Gain_X, Gain_Y,
Template_Shape_Status,Template_Rad_X, Template_Rad_Y ;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
      : TForm(Owner)
{
```

```
}
void __fastcall TForm4::ReadSetupFile(TObject *Sender)
{
FILE*settings;
char line_buffer[100];
settings = fopen("C:\\Resources\\settings\\CPsetup.txt","r");
fgets(line_buffer,90,settings);  // read line 1 from the settings file;
Subject_ID = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 2 from the settings file;
Dead_Zone_Shape_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 4  from settings file;
Dead_Zone_Rad_X = atoi(line_buffer);  //Dead zone radius Direction, 0 to 2048
fgets(line_buffer,90,settings);  //read line 5 from settings file;
Dead_Zone_Rad_Y = atoi(line_buffer);  //Dead zone radius Speed, 0 to 2048
fgets(line_buffer,90,settings);  // read line 6  from settings file;
Template_Shape_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 7  from settings file;
Template_Rad_X = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 7  from settings file;
Template_Rad_Y = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 9 from settings file;
Bias_Axis_Enabled_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 10 from settings file;
Bias_Axis_Angle = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 11 from settings file;
Gain_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 12 from settings file;
Gain_X = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 13 from settings file;
Gain_Y = atoi(line_buffer);
fclose(settings);
}
//---------------------------------------------------------------------------
void canvas_draw_ontarget()
{
//Karl / Harshal Code ____________________________________________________________
        float DIG_zero = 2048.0;
        float dir, spd;
        float angle, L1, L2;
        float one, two;
        /* for the bias axes */
        float dir_tmp;      //prevents input dir from being overwritten during rotation
        double BiasAngleDeg;
        /* if (DEBUG_lo) printf("BiasAngleDeg = %.4d\n", BiasAngleDeg); */
        /* for the dead zone */
        int DzoneFlag = 0;                  /* 1=inside deadzone, 0=outside deadzone */
        float DZangle;                      /* for rectangular deadzone, angle to vertex */
        float DZ_x, DZ_y;                   /* local x- and y-boundary values */
        /* for the template */
        int TP3XRad = 2000;
        int TP3YRad = 2000;
        int TempFlag = 0;                   /* 1=outside template, 0=inside template */
        int Td[4];
        float TMPT_x = 0;
        float TMPT_y = 0;                   /* local x- and y-boundary values
```

```
/* Debugging */
int iStatus = 0;
int iStatusReturn = 0;                  /* 0=good, 1=deadzone violation, 2+=template violation */

/***** APPLY THE ALGORITHM *****/
/* offset the axes so that (0,0) marks the origin */
dir = (float)direction1 - DIG_zero;
spd = (float)speed1 - DIG_zero;

/* Bias Axis Adjustment */
/* moves both the speed and direction axes together maintains internal
   90 degree alignment */
if(Bias_Axis_Enabled_Status)
{
        BiasAngleDeg = Bias_Axis_Angle * Pi/180.0;
        dir_tmp = dir;
        dir = (float)((cos(BiasAngleDeg))*dir_tmp - (sin(BiasAngleDeg))*spd);
        spd = (float)((sin(BiasAngleDeg))*dir_tmp + (cos(BiasAngleDeg))*spd);
}
/* Deadzone Determination */
/* find input magnitude squared */
L1 = dir*dir + spd*spd;
 /* ellipse or rectangle */
switch (Dead_Zone_Shape_Status)
{
        case 0:
                DZ_x = 0;
                DZ_y = 0;
                break;
        case 1:         /* elliptical deadzone */
                if (dir != 0)
                {       /* avoid dividing by 0 */
                        /* find corresponding point on deadzone */
                        L2 = spd/dir;
                        angle = (float)atan2(Dead_Zone_Rad_X*L2, Dead_Zone_Rad_Y);
                        one = (float)(Dead_Zone_Rad_X*cos(angle));
                        two = (float)(Dead_Zone_Rad_Y*sin(angle));
                        L2 = (one*one) + (two*two);
                        /* if within deadzone, set the flag; otherwise save intersection points */
                        if (L1 < L2) DzoneFlag = 1;
                        else {
                                angle = (float)atan2(spd,dir);
                                DZ_x = (float)(sqrt(L2)*cos(angle));
                                DZ_y = (float)(sqrt(L2)*sin(angle));
                        }
                }
                else
                {       /* dir == 0 */
                        /* if within deadzone, set the flag; otherwise find intersection
                           points */
                        if (fabs(spd) < Dead_Zone_Rad_Y) DzoneFlag = 1;
                        else
                        {
                        DZ_x = 0.0f;
                        DZ_y = (float)(spd>0? Dead_Zone_Rad_Y:-Dead_Zone_Rad_Y);
                        }
```

```
        }
        break;

case 2:         /* rectangular deadzone */
        /* if within deadzone, set the flag; otherwise find intersection points */
        if ((fabs(dir) <= Dead_Zone_Rad_X) & (fabs(spd) <= Dead_Zone_Rad_Y))
                DzoneFlag = 1;
        else
        {
        DZangle = atan2((float)Dead_Zone_Rad_Y, (float)Dead_Zone_Rad_X);
                if ((spd != 0) & (dir != 0))
                {       /* avoid dividing by 0 */
                        L2 = (float)atan2(spd, dir);
                if (((L2>=0)&(L2<=DZangle)) || ((L2<=0)&(L2>=-DZangle)))
                {
                        DZ_x = (float)Dead_Zone_Rad_X;
                        DZ_y = (float)(tan(L2)*Dead_Zone_Rad_X);
                }
                else if ((L2>DZangle) & (L2<=Pi-DZangle))
                {
                        DZ_x = (float)(Dead_Zone_Rad_Y/tan(L2));
                        DZ_y = (float)Dead_Zone_Rad_Y;
                }
                else if ((L2<-DZangle) & (L2>=DZangle-Pi))
                {
                        DZ_x = (float)(-Dead_Zone_Rad_Y/tan(L2));
                        DZ_y = (float)-Dead_Zone_Rad_Y;
                }
                else {
                        DZ_x = (float)-Dead_Zone_Rad_X;
                        DZ_y = (float)(-tan(L2)*Dead_Zone_Rad_X);
                }
        }
        else
        {       /* spd or dir == 0 */
                if ((spd==0) & (dir>=0))
                {
                        DZ_x = (float)Dead_Zone_Rad_X;
                        DZ_y = 0.0f;
                }
                else if ((spd==0) & (dir<0))
                {
                        DZ_x = (float)-Dead_Zone_Rad_X;
                        DZ_y = 0.0f;
                }
                else if ((dir==0) & (spd>=0))
                {
                        DZ_x = 0.0f;
                        DZ_y = (float)Dead_Zone_Rad_Y;
                }
                else
                {
                        DZ_x = 0.0f;
                        DZ_y = (float)-Dead_Zone_Rad_Y;
                }
        }
```

```
		}
		break;
}

/* if inside the deadzone, set inputs to 0 */
if (DzoneFlag)
{
		dir = 0;
		spd = 0;
		DZ_x = 0;
		DZ_y = 0;
		TMPT_x = 0;
		TMPT_y = 0;
		iStatusReturn = 1;
}
/* Ensure smooth transition out of deadzone */
if (!DzoneFlag)
{
		dir = dir - DZ_x;
		spd = spd - DZ_y;
}
/* Apply Gain */
if (!DzoneFlag)
{ /* if outside the deadzone */
		dir = Gain_X*dir;
		spd = Gain_Y*spd;
}
/* Apply Template */
if (!DzoneFlag)
{		/* apply only if not in deadzone */
		/* recompute input magnitude */
		L1 = dir*dir + spd*spd;
		switch (Template_Shape_Status)
		{
			case 0:
				break;
			case 1:	/* elliptical template */
				if(dir != 0)
				{	/* avoid dividing by 0 */
					/* find corresponding point on template in QI and QIV */
					L2 = spd/dir;
					angle = (float)atan2(Template_Rad_X*L2, Template_Rad_Y);
					one = (float)(Template_Rad_X*cos(angle));
					two = (float)(Template_Rad_Y*sin(angle));
					L2 = (one*one) + (two*two);
					/* save location; correct quandrant */
					angle = (float)(atan2(spd, dir));
					TMPT_x = (float)(sqrt(L2)*cos(angle));
					TMPT_y = (float)(sqrt(L2)*sin(angle));
					/* if outside the template, set input to template value */
					if (L1 > L2)
					{
						TempFlag = 1;
						dir = TMPT_x;
						spd = TMPT_y;
						iStatusReturn = 2;
```

```
                }

        }
        else
        {       /* dir == 0 */
                /* determine where signal would intersect template */
        TMPT_x = 0.0f;
        TMPT_y = (float)(spd>0? Template_Rad_Y:-Template_Rad_Y);
                /* if outside the template, set input to template value */
                if (fabs(spd) > Template_Rad_Y)
                {
                        TempFlag = 1;
                        dir = 0;
                        spd = TMPT_y;
                        iStatusReturn = 3;
                }
        }
        break;
case 2:         /* astroid template */
        /* points to center x and y */
        if (dir != 0)
        {       /* avoid dividing by 0 */
                L2 = spd/dir;
                angle = (float)atan2((powl((float)Template_Rad_X*
                   fabs(L2)/(float)(Template_Rad_Y), 0.3333)), 1.0f);
          one=(float)(Template_Rad_X*cos(angle)*cos(angle)*cos(angle));
          two=(float)(Template_Rad_Y*sin(angle)*sin(angle)*sin(angle));
                L2 = one*one + two*two;
                /* if the others were wrong, should i trust this, too? */
                angle = (float)atan2(spd, dir);
                TMPT_x = (float)(sqrt(L2)*cos(angle));
                TMPT_y = (float)(sqrt(L2)*sin(angle));
                /* if outside the template, set input to template value */
                if (L1 > L2) {
                        TempFlag = 1;
                        dir = TMPT_x;
                        spd = TMPT_y;
                        iStatusReturn = 2;
                }
        }
        else
        {       /* dir == 0 */
        TMPT_x = 0.0f;
        TMPT_y = (float)(spd>0? Template_Rad_Y:-Template_Rad_Y);
                if (fabs(spd) > Template_Rad_Y)
                {
                        TempFlag = 1;
                        dir = TMPT_x;
                        spd = TMPT_y;
                        iStatusReturn = 3;
                }
        }
        break;

case 3:         /* diamond template */
        /* x- and y-radius */
```

```
                                        TP3XRad = Template_Rad_X;
                                        TP3YRad = Template_Rad_Y;
                                        L2 = (float)TP3YRad/TP3XRad;
                                        Td[0] = spd - (L2*dir - TP3YRad);
                                        Td[1] = (-L2*dir +TP3YRad) - spd;
                                        Td[2] = (L2*dir + TP3YRad) - spd;
                                        Td[3] = spd - (-L2*dir -TP3YRad);
                                        if (dir != 0)
                                         {
                                                L1 = fabs(spd/dir);
                                                if ((spd >= 0) && (dir >= 0))
                                                {
                                                        TMPT_x = (TP3YRad/(L1+L2));
                                                        TMPT_y = (L1*TP3YRad/(L1+L2));
                                                }
                                                else if ((dir <= 0) && (spd >= 0))
                                                {
                                                        TMPT_x = (TP3YRad/(-L1-L2));
                                                        TMPT_y = (L1*TP3YRad/(L1+L2));
                                                }
                                                else if ((dir<=0) && (spd<=0))
                                                {
                                                        TMPT_x = (-TP3YRad/(L1+L2));
                                                        TMPT_y = (-TP3YRad*L1/(L1+L2));
                                                }
                                                else
                                                {
                                                        TMPT_x = (TP3YRad/(L1+L2));
                                                        TMPT_y = (-TP3YRad*L1/(L1+L2));
                                                }
                                        }
                                        else
                                         {
                                                TMPT_x = 0.0f;
                                                TMPT_y = (float)(spd>=0? TP3YRad:-TP3YRad);
                                        }
                                        if (!(Td[0]>=0 && Td[1]>=0 && Td[2]>=0 && Td[3]>=0))
                                        {
                                                TempFlag = 1;
                                                dir = TMPT_x;
                                                spd = TMPT_y;
                                                iStatusReturn = 2;
                                        }
                                        break;
                                }
                        }
//End Karl / Harshal Code _____________________________________________________________
        static float tmp_xplot=350;
        static float tmp_yplot=350;
        xplot=350+(dir);
        yplot=350-(spd);
        if (xplot >= 703) xplot = 703;
        if (xplot <= 0) xplot = 0;
        if (yplot >= 703) yplot = 703;
        if (yplot <= 0) yplot = 0;
        Form4->Canvas->Brush->Color = clAqua;
```

```
        Form4->Canvas->Pen->Color = clAqua;
        Form4->Canvas->Pen->Width = 5;
        Form4->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
        fprintf(fp, "%d %d %d %d %d %d %d %d %d %d\n", direction1, speed1, xplot, yplot, xtarget, ytarget,
            miss+(hit-1), miss, hit-1, t);
        }
void draw_target(TColor cld, int x, int y)
{
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form4->Canvas->Brush->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form4->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = cld;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
        }
void draw_target_smile(TColor cld, int x, int y)
{
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form4->Canvas->Brush->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form4->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form4->Canvas->Brush->Color = clWhite;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = cld;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
}
void draw_target_frown(TColor cld, int x, int y)
{
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form4->Canvas->Brush->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form4->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.1),x+(radius*0.6),y+(radius*0.7));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = cld;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.4),x+(radius*0.6),y+(radius*0.75));
}
int inside()
```

```
{
	int timein;
	if (flag==1)
	{
		trialtime++;
		 if ((xplot-xtarget)*(xplot-xtarget)+(yplot-ytarget)*(yplot-ytarget)<radius*radius)
		 {	//inside the target
			flag1=1;
			count++;
			timein=10*count;
			Form4->Canvas->Brush->Color = clRed;
			Form4->Canvas->Pen->Color = clRed;
			Form4->Canvas->Pen->Width = 5;
			Form4->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
		 }
		 else
		 {	//outside the target
			flag1=0;
			count=0;
			Form4->Canvas->Brush->Color = clBlue;
			Form4->Canvas->Pen->Color = clBlue;
			 Form4->Canvas->Pen->Width = 5;
			Form4->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
		 }
	if (flag1==1 & timein==2000)
		{
			 draw_target(clRed, xtarget, ytarget);
			trialend=1;
			trialtime=0;
			first=0;
			return 1;
		}
	else if (flag1==0 & trialtime*10>=20000)
		{
			draw_target(clYellow, xtarget, ytarget);
			trialend=2;
			trialtime=0;
			first=0;
			return 1;
		}
	else return 0;
	}
	else return 0;
}
void generate_target()
{
		Form4->Refresh();
		randomize();
		int position=8;
		t=random(position);
		xtarget= (Holes[t].x);
		ytarget= (Holes[t].y);
		xplot = 350;
		yplot = 350;
		draw_target(clYellow, xtarget, ytarget);
}
```

```
//---------------------------------------------------------------------------
//Click Button 1
void __fastcall TForm4::Button1Click(TObject *Sender)
{
      Form4->Refresh();
      generate_target();
      fp= fopen("C:\\Resources\\Data\\CCVisit1.txt","a");
      hit=1;
      miss=0;
      Timer1->Enabled=true;
      Button1->Enabled=false;
      flag=1;
      unsigned int count1=0;
      int ab11, ab21;
// Serial Port Communication
      DCB dcbCommPort;
      hComm=CreateFile("COM1",GENERIC_READ|GENERIC_WRITE,0,NULL,
          OPEN_EXISTING,0,NULL);
      GetCommState(hComm,&dcbCommPort);
      BuildCommDCB("38400,N,8,1", &dcbCommPort);
      SetCommState(hComm,&dcbCommPort);
      SetCommMask(hComm,EV_RXCHAR);
      GetCommTimeouts(hComm,&ctmoNew);
      ctmoNew.ReadIntervalTimeout=2.7;
      ctmoNew.ReadTotalTimeoutConstant=0;
      ctmoNew.ReadTotalTimeoutMultiplier=0;
      SetCommTimeouts(hComm,&ctmoNew);
// Get start time
      gettime(&bt);
      while(1)
      {
            ClearCommError(hComm,&dwError,&cs);
            unsigned char InBuff[10];
            DWORD dwBytesRead;
            Application->ProcessMessages();
            while(1)
            {
                  if (WaitCommEvent(hComm,&dwEvent,NULL))
                        if (dwEvent & EV_RXCHAR)
                        {
                              ClearCommError(hComm,&dwError,&cs);
                              ReadFile(hComm,InBuff,4,&dwBytesRead,NULL);
                              ClearCommError(hComm,&dwError,&cs);
                              ab11=(unsigned int)(unsigned char)(InBuff[0]);
                              if ((ab11&240)!=16) break;
                              ab21=(unsigned int)(unsigned char)(InBuff[2]);
                              if ((ab21&240)!=32) break;
                              direction1=data1[count1*2]=(ab11&15)*256+(unsigned int)(unsigned
                                char)(InBuff[1]);
                              speed1=data1[count1*2+1]=(ab21&15)*256+(unsigned int)(unsigned
                                char)(InBuff[3]);
                              ++count1;
                              break;
                         }
             }
            gettime(&et);
```

```
            if ((miss+hit)>=33)
            {
            fprintf(fp, "%d %d %d %d %d %d %d %d %d %d %d\n", direction1, speed1, xplot, yplot,
                    xtarget, ytarget, miss+(hit-1), miss, hit-1, t, Stamp);
            break;
            }
        }
        Timer1->Enabled=false;
        SetCommMask(hComm,0);
        PurgeComm(hComm,PURGE_RXABORT);
        SetCommTimeouts(hComm,&ctmoOld);
        CloseHandle(hComm);
        Button1->Enabled=true;
}
//---------------------------------------------------------------------------
//Timer
void __fastcall TForm4::Timer1Timer(TObject *Sender)
{
    char strBuffer[9];
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", bt.ti_hour, bt.ti_min, bt.ti_sec, bt.ti_hund);
    Edit3->Text=strBuffer;
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", et.ti_hour, et.ti_min, et.ti_sec, et.ti_hund);
    Edit4->Text=strBuffer;
    Edit1->Text=IntToStr(direction1);
    Edit2->Text=IntToStr(speed1);
    Edit5->Text=IntToStr(hit-1);
    Edit6->Text=IntToStr(miss);
    Edit7->Text=IntToStr(t+1);
    Edit8->Text=IntToStr(Stamp);
    if (flag==1 & !inside() & !trialend) canvas_draw_ontarget();
    if (trialend==1 & first==0)
    {
            Timer1->Interval=1000;
            draw_target_smile(clRed,xtarget,ytarget);
            aflag=1;
            first=1;
            hit=hit+1;
            Edit5->Text=IntToStr(hit-1);
    }
    else if (trialend==2 & first==0)
    {
            Timer1->Interval=1000;
            draw_target_frown(clLime,xtarget,ytarget);
            aflag=1;
            first=1;
            miss=miss+1;
            Edit6->Text=IntToStr(miss);
    }
    if (trialend!=0 & aflag==0)
    {
            generate_target();
            Timer1->Interval=10;
            trialend=0;
    }
    if (first==1) aflag=0;
}
```

```
void __fastcall TForm4::Form4Paint(TObject *Sender)
{
        Canvas->Brush->Color = clBlack;
        Canvas->Brush->Color = clWhite;
        Canvas->Pen->Color = clBlack;
        Canvas->Pen->Width = 3;
        Canvas->Ellipse(350-300,350-300,350+300,350+300);
        Canvas->Ellipse(350-150,350-150,350+150,350+150);
        Canvas->Brush->Color = clBlack;
        Canvas->Pen->Color = clBlack;
        Canvas->Ellipse(350-5,350-5,350+5,350+5);
        Canvas->Pen->Color = clBlack;
        Canvas->Ellipse(350-7,350-7,350+7,350+7);
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button2Click(TObject *Sender)
{
        //close
        //Form1->Visible=true;
        Form4->Visible=false;
        fclose(fp);
        Form4->Close();
}
//---------------------------------------------------------------------------
//Scanning Bar
void __fastcall TForm4::Timer2Timer(TObject *Sender)
{
// If the current color is red scan to green
   if( shpRed->Brush->Color == clRed)
   {
   // Change the color to green
      Timer2->Interval       = 2000;
      shpRed->Brush->Color    = clBlack;
      shpYellow->Brush->Color = clBlack;
      shpGreen->Brush->Color  = clGreen;
      shpBlue->Brush->Color   = clBlack;
   }
   //If the color is green scan to yellow
   else if( shpGreen->Brush->Color == clGreen)
   {
      // Change the color to yellow
      Timer2->Interval       = 2000;
      shpRed->Brush->Color    = clBlack;
      shpYellow->Brush->Color = clYellow;
      shpGreen->Brush->Color  = clBlack;
      shpBlue->Brush->Color   = clBlack;
   }
   //If the color is yellow scan to blue
   else if( shpYellow->Brush->Color == clYellow)
   {
      // Change the color to Blue
      Timer2->Interval       = 2000;
      shpRed->Brush->Color    = clBlack;
      shpYellow->Brush->Color = clBlack;
      shpGreen->Brush->Color  = clBlack;
      shpBlue->Brush->Color   = clBlue;
```

```
    }
    //If the color is Blue scan to Red
    else if (shpBlue->Brush->Color   == clBlue)
    {
       Timer2->Interval        = 2000;
       shpRed->Brush->Color    = clRed;
       shpYellow->Brush->Color = clBlack;
       shpGreen->Brush->Color  = clBlack;
       shpBlue->Brush->Color   = clBlack;
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button3Click(TObject *Sender)
{
Stamp = Stamp + 1;
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
//Customized Control Visit 2
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#define PI 3.1415926
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CGAUGES"
#pragma resource "*.dfm"
TPoint Holes[8] = {TPoint(350, 200),TPoint(500, 350),TPoint(350, 500),
                TPoint(200, 350),TPoint(350, 50),TPoint(650, 350),
                TPoint(350, 650),TPoint(50, 350)};
TForm4 *Form4;
FILE*settings;
FILE*fp;
//---------------------------------------------------------------------------
//radius of 96/2 = 48 = 1 inch (96 pixel) diameter target
int radius = 48;
AnsiString subject_id;
HANDLE hComm=NULL;
COMMTIMEOUTS ctmoNew={0},ctmoOld;
DWORD dwBytesRead;
DWORD dwEvent,dwError;
COMSTAT cs;
struct time bt, et;
int scan; //1=forward, 2=back, 3=left, 4=right
int scanstop = 0; //0=scan, 1=pause scanning
int count1;
int direction1, speed1, dir, spd;
int data1[400000];
int scale=4;
int stamp = 0;
//initial position of the axquired target
int xtarget, ytarget;
//cursor position
int xplot=350, yplot=350, hit =1 , miss=0, t;
int flag=0, flag1=0, count=0, trialend=0, trialtime=0, first=0, aflag=0;
int p0=0, p1=0, p2=0, p3=0, p4=0, p5=0, p6=0, p7=0;
int canvasflag =1;
int Subject_ID, Dead_Zone_Shape_Status, Dead_Zone_Rad_X, Dead_Zone_Rad_Y,
Bias_Axis_Enabled_Status, Bias_Axis_Angle, Gain_Status, Gain_X, Gain_Y,
Template_Shape_Status,Template_Rad_X, Template_Rad_Y ;
int SwitchType, Option1, Force1, Direction1, Option2, Force2, Direction2;
int c=0;
int activate = 2100, moving=0, move_d=0, activescan=1, activate_scan;
int DZ_x, DZ_y;      /* local x- and y-boundary values */
//float spd, dir;
```

```
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm4::ReadSetupFile(TObject *Sender)
{
//Import tuning software settings
FILE*settings;
char line_buffer[100];
settings = fopen("C:\\Resources\\settings\\CPsetup.txt","r");
fgets(line_buffer,90,settings);  // read line 1 from the settings file;
Subject_ID = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 2 from the settings file;
Dead_Zone_Shape_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 4  from settings file;
Dead_Zone_Rad_X = atoi(line_buffer);  //Dead zone radius Direction, 0 to 2048
fgets(line_buffer,90,settings);  //read line 5 from settings file;
Dead_Zone_Rad_Y = atoi(line_buffer);  //Dead zone radius Speed, 0 to 2048
fgets(line_buffer,90,settings);  // read line 6  from settings file;
Template_Shape_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 7  from settings file;
Template_Rad_X = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 7  from settings file;
Template_Rad_Y = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 9 from settings file;
Bias_Axis_Enabled_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 10 from settings file;
Bias_Axis_Angle = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 11 from settings file;
Gain_Status = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 12 from settings file;
Gain_X = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 13 from settings file;
Gain_Y = atoi(line_buffer);
fclose(settings);
//Import switch settings
FILE*switchsettings;
settings = fopen("C:\\Resources\\settings\\switch.txt","r");
fgets(line_buffer,90,settings);  // read line 1 from the settings file;
Subject_ID = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 2 from settings file;
SwitchType = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 3 from settings file;
Option1 = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 4 from settings file;
Force1 = atoi(line_buffer);      //or digital value????
fgets(line_buffer,90,settings);  // read line 5 from settings file;
Direction1 = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 6 from settings file;
Option2 = atoi(line_buffer);
fgets(line_buffer,90,settings);  // read line 7 from settings file;
Force2 = atoi(line_buffer);      //or digital value????
fgets(line_buffer,90,settings);  // read line 8 from settings file;
Direction2 = atoi(line_buffer);
fclose(switchsettings);
```

```
}
//---------------------------------------------------------------------------
void canvas_draw_ontarget()
{
//Apply Tuning Parameters to Joystick for initial Customization
        float DIG_zero = 2048.0;
        //float dir, spd;
        float angle, L1, L2;
        float one, two;
        /* for the bias axes */
        float dir_tmp;      //prevents input dir from being overwritten during rotation
        double BiasAngleDeg;
        /* if (DEBUG_lo) printf ("BiasAngleDeg = %.4d\n", BiasAngleDeg); */
        /* for the dead zone */
        int DzoneFlag = 0;     /* 1=inside deadzone, 0=outside deadzone */
        float DZangle;          /* for rectangular deadzone, angle to vertex */
        /* for the template */
        int TP3XRad = 2000;
        int TP3YRad = 2000;
        int TempFlag = 0;      /* 1=outside template, 0=inside template */
        int Td[4];
        float TMPT_x = 0;
        float TMPT_y = 0;      /* local x- and y-boundary values
        /* Debugging */
        int iStatus = 0;
        int iStatusReturn = 0; /* 0=good, 1=deadzone violation, 2+=template violation */

/***** APPLY THE ALGORITHM *****/
        /* offset the axes so that (0,0) marks the origin */
        dir = (float)direction1 - DIG_zero;
        spd = (float)speed1 - DIG_zero;
        /* Bias Axis Adjustment */
        /* moves both the speed and direction axes together maintains internal
          90 degree alignment */
        if(Bias_Axis_Enabled_Status)
        {
                BiasAngleDeg = Bias_Axis_Angle * Pi/180.0;
                dir_tmp = dir;
                dir = (float)((cos(BiasAngleDeg))*dir_tmp - (sin(BiasAngleDeg))*spd);
                spd = (float)((sin(BiasAngleDeg))*dir_tmp + (cos(BiasAngleDeg))*spd);
        }
        /* Deadzone Determination */
        /* find input magnitude squared */
        L1 = dir*dir + spd*spd;
        /* ellipse or rectangle */
        switch (Dead_Zone_Shape_Status)
        {
                case 0:
                        DZ_x = 0;
                        DZ_y = 0;
                        break;
                case 1:         /* elliptical deadzone */
                        if (dir != 0)
                        {       /* avoid dividing by 0 */
                                /* find corresponding point on deadzone */
                                L2 = spd/dir;
```

```
                angle = (float)atan2(Dead_Zone_Rad_X*L2, Dead_Zone_Rad_Y);
                one = (float)(Dead_Zone_Rad_X*cos(angle));
                two = (float)(Dead_Zone_Rad_Y*sin(angle));
                L2 = (one*one) + (two*two);
                /* thought about quadrant needs to go here */
                /* if within deadzone, set the flag; otherwise save intersection points */
                if (L1 < L2) DzoneFlag = 1;
                else
                {
                        angle = (float)atan2(spd,dir);
                        DZ_x = (float)(sqrt(L2)*cos(angle));
                        DZ_y = (float)(sqrt(L2)*sin(angle));
                }
        }
        else
        {       /* dir == 0 */
                /* if within deadzone, set the flag; otherwise find intersection
                   points */
                if (fabs(spd) < Dead_Zone_Rad_Y) DzoneFlag = 1;
                else
                {
                DZ_x = 0.0f;
                DZ_y = (float)(spd>0? Dead_Zone_Rad_Y:-Dead_Zone_Rad_Y);
                }
        }
        break;
case 2:         /* rectangular deadzone */
        /* if within deadzone, set the flag; otherwise find intersection  points */
        if ((fabs(dir) <= Dead_Zone_Rad_X) & (fabs(spd) <= Dead_Zone_Rad_Y))
                DzoneFlag = 1;
        else
        {
        DZangle = atan2((float)Dead_Zone_Rad_Y, (float)Dead_Zone_Rad_X);
                if ((spd != 0) & (dir != 0))
                {       /* avoid dividing by 0 */
                        L2 = (float)atan2(spd, dir);
                if (((L2>=0)&(L2<=DZangle)) || ((L2<=0)&(L2>=-DZangle)))
                {
                        DZ_x = (float)Dead_Zone_Rad_X;
                        DZ_y = (float)(tan(L2)*Dead_Zone_Rad_X);
                        }
                        else if ((L2>DZangle) & (L2<=Pi-DZangle))
                        {
                        DZ_x = (float)(Dead_Zone_Rad_Y/tan(L2));
                        DZ_y = (float)Dead_Zone_Rad_Y;
                        }
                        else if ((L2<-DZangle) & (L2>=DZangle-Pi))
                        {
                        DZ_x = (float)(-Dead_Zone_Rad_Y/tan(L2));
                        DZ_y = (float)-Dead_Zone_Rad_Y;
                        }
                        else
                        {
                        DZ_x = (float)-Dead_Zone_Rad_X;
                        DZ_y = (float)(-tan(L2)*Dead_Zone_Rad_X);
                        }
```

```
                        }
                        else
                        {       /* spd or dir == 0 */
                                if ((spd==0) & (dir>=0))
                                {
                                        DZ_x = (float)Dead_Zone_Rad_X;
                                        DZ_y = 0.0f;
                                }
                                else if ((spd==0) & (dir<0))
                                {
                                        DZ_x = (float)-Dead_Zone_Rad_X;
                                        DZ_y = 0.0f;
                                }
                                else if ((dir==0) & (spd>=0))
                                {
                                        DZ_x = 0.0f;
                                        DZ_y = (float)Dead_Zone_Rad_Y;
                                }
                                else
                                {
                                        DZ_x = 0.0f;
                                        DZ_y = (float)-Dead_Zone_Rad_Y;
                                }
                        }
                }
                break;
}

/* if inside the deadzone, set inputs to 0 */
if (DzoneFlag)
{
        dir = 0;
        spd = 0;
        DZ_x = 0;
        DZ_y = 0;
        TMPT_x = 0;
        TMPT_y = 0;
        iStatusReturn = 1;
}
/* Ensure smooth transition out of deadzone */
if (!DzoneFlag)
{
        dir = dir - DZ_x;
        spd = spd - DZ_y;
}
/* Apply Gain */
if (!DzoneFlag)
{
        /* if outside the deadzone */
        dir = Gain_X*dir;
        spd = Gain_Y*spd;
}
/* Apply Template */
if (!DzoneFlag)
{       /* apply only if not in deadzone */
        L1 = dir*dir + spd*spd;
```

```
switch (Template_Shape_Status)
{
        case 0:
                break;
        case 1:  /* elliptical template */
                if(dir != 0)
                {       /* avoid dividing by 0 */
                        /* find corresponding point on template in QI and QIV */
                        L2 = spd/dir;
                        angle = (float)atan2(Template_Rad_X*L2, Template_Rad_Y);
                        one = (float)(Template_Rad_X*cos(angle));
                        two = (float)(Template_Rad_Y*sin(angle));
                        L2 = (one*one) + (two*two);
                        /* save location; correct quandrant */
                        angle = (float)(atan2(spd, dir));
                        TMPT_x = (float)(sqrt(L2)*cos(angle));
                        TMPT_y = (float)(sqrt(L2)*sin(angle));
                        /* if outside the template, set input to template value */
                        if (L1 > L2) {
                                TempFlag = 1;
                                dir = TMPT_x;
                                spd = TMPT_y;
                                iStatusReturn = 2;
                        }
                }
                else
                {       /* dir == 0 */
                        /* determine where signal would intersect template */
                TMPT_x = 0.0f;
                TMPT_y = (float)(spd>0? Template_Rad_Y:-Template_Rad_Y);
                        /* if outside the template, set input to template value */
                        if (fabs(spd) > Template_Rad_Y)
                        {
                                TempFlag = 1;
                                dir = 0;
                                spd = TMPT_y;
                                iStatusReturn = 3;
                        }
                }
                break;
        case 2:  /* astroid template */
                /* points to center x and y */
                if (dir != 0)
                {       /* avoid dividing by 0 */
                        L2 = spd/dir;
                        angle = (float)atan2((powl((float)Template_Rad_X*
                                fabs(L2)/(float)(Template_Rad_Y), 0.3333)), 1.0f);
                  one=(float)(Template_Rad_X*cos(angle)*cos(angle)*cos(angle));
                  two=(float)(Template_Rad_Y*sin(angle)*sin(angle)*sin(angle));
                        L2 = one*one + two*two;
                        angle = (float)atan2(spd, dir);
                        TMPT_x = (float)(sqrt(L2)*cos(angle));
                        TMPT_y = (float)(sqrt(L2)*sin(angle));
                        /* if outside the template, set input to template value */
```

```
                if (L1 > L2)
                {
                        TempFlag = 1;
                        dir = TMPT_x;
                        spd = TMPT_y;
                        iStatusReturn = 2;
                }
        }
        else
        {       /* dir == 0 */
        TMPT_x = 0.0f;
        TMPT_y = (float)(spd>0? Template_Rad_Y:-Template_Rad_Y);
                if (fabs(spd) > Template_Rad_Y)
                {
                        TempFlag = 1;
                        dir = TMPT_x;
                        spd = TMPT_y;
                        iStatusReturn = 3;
                }
        }
        break;

case 3: /* diamond template */
        /* x- and y-radius */
        TP3XRad = Template_Rad_X;
        TP3YRad = Template_Rad_Y;
        L2 = (float)TP3YRad/TP3XRad;
        Td[0] = spd - (L2*dir - TP3YRad);
        Td[1] = (-L2*dir +TP3YRad) - spd;
        Td[2] = (L2*dir + TP3YRad) - spd;
        Td[3] = spd - (-L2*dir -TP3YRad);
        if (dir != 0)
        {
                L1 = fabs(spd/dir);
                if ((spd >= 0) && (dir >= 0))
                {
                        TMPT_x = (TP3YRad/(L1+L2));
                        TMPT_y = (L1*TP3YRad/(L1+L2));
                }
                else if ((dir <= 0) && (spd >= 0))
                {
                        TMPT_x = (TP3YRad/(-L1-L2));
                        TMPT_y = (L1*TP3YRad/(L1+L2));
                }
                else if ((dir<=0) && (spd<=0))
                {
                        TMPT_x = (-TP3YRad/(L1+L2));
                        TMPT_y = (-TP3YRad*L1/(L1+L2));
                }
                else
                {
                        TMPT_x = (TP3YRad/(L1+L2));
                        TMPT_y = (-TP3YRad*L1/(L1+L2));
                }
        }
```

```
                                        else
                                        {
                                                TMPT_x = 0.0f;
                                                TMPT_y = (float)(spd>=0? TP3YRad:-TP3YRad);
                                        }
                                        if (!(Td[0]>=0 && Td[1]>=0 && Td[2]>=0 && Td[3]>=0))
                                        {
                                                TempFlag = 1;
                                                dir = TMPT_x;
                                                spd = TMPT_y;
                                                iStatusReturn = 2;
                                        }
                                        break;
                        }
                }
//Switch Control Options ------------------------------------------------------------
//Proportional Control, No Switch, No Scanning --------------------------------
        if (SwitchType == 1)
        {
        xplot=xplot+(dir/100);
        yplot=yplot-(spd/100);
        if (xplot >= 800) xplot = 800;
        if (xplot <= 0) xplot = 0;
        if (yplot >= 800) yplot = 800;
        if (yplot <= 0) yplot = 0;
        }
//All Directions,input in any direction will start or stop movement------------
        if (SwitchType == 2)
        {
                if (canvasflag==1)
                {
                  moving = 0;
                  move_d = 0;
                  direction1=2048;
                  speed1=2048;
                  canvasflag=0;
                }
                //Up and Down
                if ((fabs(spd) < Force1 & fabs(dir) < Force1)& move_d == 0)
                {moving =0;}
                if ((fabs(spd))>= Force1 & moving ==0)
                {moving = 1;
                if (spd>0) {move_d = 1;}
                else if (spd<0){move_d = 3;}
                }
                else if (fabs(spd)< Force1 & moving ==1)
                {moving = 2;}
                else if (fabs(spd)>= Force1 & moving ==2)
                {moving = 3;}
                else if (fabs(spd)< Force1 & moving ==3)
                {moving = 0;}
                //Left and right
                if ((fabs(dir))>= Force1 & moving ==0)
                {moving = 1;
                if (dir>0) {move_d = 2;}
                else if (dir<0){move_d = 4;}
```

```
            }
            else if (fabs(dir)< Force1 & moving ==1)
            {moving = 2;}
            else if (fabs(dir)>= Force1 & moving ==2)
            {moving = 3;}
            else if  (fabs(dir)< Force1 & moving ==3)
            {moving = 0;}
            if (moving == 2 & move_d == 1)
            {yplot=yplot - 1; xplot=xplot;}
            if (moving == 2 & move_d == 3)
            {yplot=yplot + 1; xplot=xplot;}
            if (moving == 2 & move_d == 2)
            {xplot=xplot + 1; yplot=yplot;}
            if (moving == 2 & move_d == 4)
            {xplot=xplot - 1; yplot=yplot;}
            if (moving == 0)
            {yplot=yplot; xplot=xplot;}
            if (moving ==0 & move_d == 0)
            {yplot=yplot; xplot=xplot;}
      if (xplot >= 800) xplot = 800;
      if (xplot <= 0) xplot = 0;
      if (yplot >= 800) yplot = 800;
      if (yplot <= 0) yplot = 0;
      }
//One Direction, Constant Force, Switch Control, Passive Scanning----------
      if (SwitchType == 3)
      {
            int shpRed, shpGreen, shpBlue, shpYellow;
            if (canvasflag==1)
                  {
                    direction1=2048;
                    speed1=2048;
                    canvasflag=0;
                  }
            if (xplot >= 800) xplot = 800;
            if (xplot <= 3) xplot = 3;
            if (yplot >= 680) yplot = 680;
            if (yplot <= 3) yplot = 3;
            xplot=xplot;
            yplot=yplot;
            //Forward
            if (speed1>= activate & scan ==4)
            {
                scanstop=0;  //pause
                yplot= yplot -1;
                xplot=xplot;
            }
            //Backward
            else if (speed1>= activate & scan ==1)
            {
                scanstop=0;
                yplot= yplot +1;
                xplot=xplot;
            }
            //Left
                else if (speed1>= Force1 && scan ==2)
```

```
                {
                    scanstop=0;
                    yplot=yplot;
                    xplot=xplot-1;
                }
                //Right
                else if (speed1>= Force1 && scan ==3)
                {
                    scanstop=0;
                     yplot= yplot;
                    xplot=xplot+1;
                }
                if (speed1< 2048)
                {
                    scanstop=1;
                    xplot=xplot;
                    yplot=yplot;
                }
}

//2 Directions, Active Scanning
      if (SwitchType == 4)
      {
      //scanstop = 2;
      int shpRed, shpGreen, shpBlue, shpYellow;
      //if (canvasflag==1)
            //{
              //direction1=2048;
              //speed1=2048;
              //canvasflag=0;
            //}
      //if (fabs(spd)>= Force2 & spd<0 & activate_scan ==0)
      //{activate_scan =1;}
      //if (fabs(spd)< Force2 & activate_scan ==1)
      //{activate_scan =0;}
      //if (fabs(spd)< Force2 & activate_scan ==1)
      //{activate_scan =0;}

      if (activescan == 1 & fabs(spd)>= Force2 & spd<0)
      {activate_scan = 1;}
      if (activescan == 1 & activate_scan ==1 & fabs(spd)< Force2)
      {activescan = 2;
      activate_scan=0;}
      if (activescan == 2 & fabs(spd)>= Force2 & spd<0)
      {activate_scan = 1;}
      if (activescan == 2 & activate_scan ==1 & fabs(spd)< Force2)
      {activescan = 3;
      activate_scan=0;}
      if (activescan == 3 & fabs(spd)>= Force2 & spd<0)
      {activate_scan = 1;}
      if (activescan == 3 & activate_scan ==1 & fabs(spd)< Force2)
      {activescan = 4;
      activate_scan=0;}
      if (activescan == 4 & fabs(spd)>= Force2 & spd<0)
      {activescan = 1;}
      if (activescan == 4 & activate_scan ==1 & fabs(spd)< Force2)
```

```
        {activescan = 1;
        activate_scan=0;}
        if (xplot >= 800) xplot = 800;
        if (xplot <= 3) xplot = 3;
        if (yplot >= 680) yplot = 680;
        if (yplot <= 3) yplot = 3;
        if ((fabs(spd))>= Force1 & spd>0 & moving ==0)
        {moving = 1;}
        if (fabs(spd)< Force1 & spd>0 & moving ==1)
        {moving = 2;}
        if (fabs(spd)>= Force1 & spd>0 & moving ==2)
        {moving = 3;}
        if (fabs(spd)< Force1 & spd>0 & moving ==3)
        {moving = 0;}
        //Forward
        if (moving ==2 & scan ==4)
        {
        yplot= yplot -1;
        xplot=xplot;
        }
        //Backward
        else if (moving==2 &scan ==1)
        {
        yplot= yplot +1;
        xplot=xplot;
        }
        //Left
        else if (moving == 2 && scan ==2)
        {
        yplot=yplot;
        xplot=xplot-1;
        }
        //Right
        else if (moving ==2 && scan ==3)
        {
        yplot= yplot;
        xplot=xplot+1;
        }
        if (moving == 0)
        {
        xplot=xplot;
        yplot=yplot;
        }
        }
//2 Directions, Passive Scanning
        if (SwitchType == 7)
        {
        int shpRed, shpGreen, shpBlue, shpYellow;
        if (canvasflag==1)
                {
                  direction1=2048;
                  speed1=2048;
                  canvasflag=0;
                }
        if (xplot >= 800) xplot = 800;
        if (xplot <= 3) xplot = 3;
```

```
        if (yplot >= 680) yplot = 680;
        if (yplot <= 3) yplot = 3;
        xplot=xplot;
        yplot=yplot;
        //Forward
        if (speed1>= activate & scan ==4)
        {
        scanstop=0;  //pause
        yplot= yplot -1;
        xplot=xplot;
        //Stop
        }
        //Backward
        else if (speed1>= activate & scan ==1)
        {
        scanstop=0;
        yplot= yplot +1;
        xplot=xplot;
        }
        //Left
        else if (speed1>= activate && scan ==2)
        {
        scanstop=0;
        yplot=yplot;
        xplot=xplot-1;
        //Stop
        }
        //Right
        else if (speed1>= activate && scan ==3)
        {
        scanstop=0;
        yplot= yplot;
        xplot=xplot+1;
        }
        if (speed1< 2048)
        {
        scanstop=1;
        xplot=xplot;
        yplot=yplot;
        }
        }

       Form4->Canvas->Brush->Color = clAqua;
        Form4->Canvas->Pen->Color = clAqua;
        Form4->Canvas->Pen->Width = 5;
        Form4->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
        fprintf(fp, "%d %d %d %d %d %d %d %d %d %d\n", direction1, speed1, xplot, yplot, xtarget, ytarget,
             miss+(hit-1), miss, hit-1, t);
}
void draw_target(TColor cld, int x, int y)
{
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form4->Canvas->Brush->Color = clBlack;
```

```
        Form4->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form4->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = cld;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
        }
void draw_target_smile(TColor cld, int x, int y)
{
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form4->Canvas->Brush->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form4->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form4->Canvas->Brush->Color = clWhite;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.7));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = cld;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.02),x+(radius*0.6),y+(radius*0.3));
}
void draw_target_frown(TColor cld, int x, int y)
{
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-radius,y-radius,x+radius,y+radius);
        Form4->Canvas->Brush->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius/2),y-(radius/2),x-((radius/2)-8),y-((radius/2)-8));
        Form4->Canvas->Ellipse(x+((radius/2)-8), y-((radius/2)-8), x+(radius/2),y-(radius/2));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = clBlack;
        Form4->Canvas->Ellipse(x-(radius * 0.6),y+(radius*0.1),x+(radius*0.6),y+(radius*0.7));
        Form4->Canvas->Brush->Color = cld;
        Form4->Canvas->Pen->Color = cld;
        Form4->Canvas->Pen->Width = 3;
        Form4->Canvas->Rectangle(x-(radius * 0.6),y+(radius*0.4),x+(radius*0.6),y+(radius*0.75));
}
int inside()
{
        int timein;
        if (flag==1)
        {
        trialtime++;
                if ((xplot-xtarget)*(xplot-xtarget)+(yplot-ytarget)*(yplot-ytarget)<radius*radius)
                {//inside the target
                flag1=1;
                count++;
                timein=10*count;
                Form4->Canvas->Brush->Color = clRed;
                Form4->Canvas->Pen->Color = clRed;
                Form4->Canvas->Pen->Width = 5;
```

```
            Form4->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
            }
            else
            {
            //outside the target
            flag1=0;
            count=0;
            Form4->Canvas->Brush->Color = clBlue;
            Form4->Canvas->Pen->Color = clBlue;
            Form4->Canvas->Pen->Width = 5;
            Form4->Canvas->Ellipse(xplot-1,yplot-1,xplot+1,yplot+1);
            }
      if (flag1==1 & timein==2000)
            {
            draw_target(clRed, xtarget, ytarget);
            trialend=1;
            trialtime=0;
            first=0;
            return 1;
            }
      else if (flag1==0 & trialtime*10>=20000)
            {
            draw_target(clYellow, xtarget, ytarget);
            trialend=2;
            trialtime=0;
            first=0;
            return 1;
            }
      else return 0;
      }
      else return 0;
}
void generate_target()
{
Form4->Refresh();
randomize();
int position=8;
t=random(position);
xtarget= (Holes[t].x);
ytarget= (Holes[t].y);
xplot = 350;
yplot = 350;
move_d = 0;
moving = 0;
draw_target(clYellow, xtarget, ytarget);
}
//---------------------------------------------------------------------------
//Click Button 1
void __fastcall TForm4::Button1Click(TObject *Sender)
{
      Form4->Refresh();
      generate_target();
      fp= fopen("C:\\Resources\\Data\\CCvisit2.txt","a");
      hit=1;
      miss=0;
      moving = 0;
```

```
        move_d=0;
        Timer1->Enabled=true;
        Button1->Enabled=false;
        flag=1;
        unsigned int count1=0;
        int ab11, ab21;
// Serial Port Communication
        DCB dcbCommPort;
        hComm=CreateFile("COM1",GENERIC_READ|GENERIC_WRITE,0,NULL,
            OPEN_EXISTING,0,NULL);
        GetCommState(hComm,&dcbCommPort);
        BuildCommDCB("38400,N,8,1", &dcbCommPort);
        SetCommState(hComm,&dcbCommPort);
        SetCommMask(hComm,EV_RXCHAR);
        GetCommTimeouts(hComm,&ctmoNew);
        ctmoNew.ReadIntervalTimeout=2.7;
        ctmoNew.ReadTotalTimeoutConstant=0;
        ctmoNew.ReadTotalTimeoutMultiplier=0;
        SetCommTimeouts(hComm,&ctmoNew);
// Get start time
        gettime(&bt);
        while(1)
        {
            ClearCommError(hComm,&dwError,&cs);
            unsigned char InBuff[10];
            DWORD dwBytesRead;
            Application->ProcessMessages();
            while(1)
            {
                if (WaitCommEvent(hComm,&dwEvent,NULL))
                    if (dwEvent & EV_RXCHAR)
                    {
                        ClearCommError(hComm,&dwError,&cs);
                        ReadFile(hComm,InBuff,4,&dwBytesRead,NULL);
                        ClearCommError(hComm,&dwError,&cs);
                        ab11=(unsigned int)(unsigned char)(InBuff[0]);
                        if ((ab11&240)!=16) break;
                        ab21=(unsigned int)(unsigned char)(InBuff[2]);
                        if ((ab21&240)!=32) break;
                        direction1=data1[count1*2]=(ab11&15)*256+(unsigned int)(unsigned
                          char)(InBuff[1]);
                        speed1=data1[count1*2+1]=(ab21&15)*256+(unsigned int)(unsigned
                          char)(InBuff[3]);
                        ++count1;
                        break;
                    }
             }
            gettime(&et);
            if ((miss+hit)>=33)
            {
            fprintf(fp, "%d %d %d %d %d %d %d %d %d %d %d\n", direction1, speed1, xplot, yplot,
                    xtarget, ytarget, miss+(hit-1), miss, hit-1, t, stamp);
            break;
            }
        }
        Timer1->Enabled=false;
```

```
        SetCommMask(hComm,0);
        PurgeComm(hComm,PURGE_RXABORT);
        SetCommTimeouts(hComm,&ctmoOld);
        CloseHandle(hComm);
        Button1->Enabled=true;
}
//---------------------------------------------------------------------------
//Timer
void __fastcall TForm4::Timer1Timer(TObject *Sender)
{
    char strBuffer[9];
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", bt.ti_hour, bt.ti_min, bt.ti_sec, bt.ti_hund);
    Edit3->Text=strBuffer;
    sprintf(strBuffer, "%2d:%02d:%02d.%02d", et.ti_hour, et.ti_min, et.ti_sec, et.ti_hund);
    Edit4->Text=strBuffer;
    Edit1->Text=FloatToStr(dir);
    Edit2->Text=FloatToStr(spd);
    Edit5->Text=IntToStr(DZ_y); //hit-1
    Edit6->Text=IntToStr(DZ_x); //miss
    Edit7->Text=IntToStr(activate_scan);
    Edit8->Text=IntToStr(activescan);   //stamp
    if (flag==1 & !inside() & !trialend) canvas_draw_ontarget();
    if (trialend==1 & first==0)
    {
      Timer1->Interval=1000;
      draw_target_smile(clRed,xtarget,ytarget);
      aflag=1;
      first=1;
      hit=hit+1;
      moving = 0;
      move_d=0;
      Edit5->Text=IntToStr(hit-1);
    }
    else if (trialend==2 & first==0)
    {
      Timer1->Interval=1000;
      draw_target_frown(clLime,xtarget,ytarget);
      aflag=1;
      first=1;
      miss=miss+1;
      moving = 0;
      move_d=0;
      Edit6->Text=IntToStr(miss);
    }
    if (trialend!=0 & aflag==0)
    {
      generate_target();
      Timer1->Interval=10;
      trialend=0;
    }
    if (first==1) aflag=0;
}
void __fastcall TForm4::Form4Paint(TObject *Sender)
{
        Canvas->Brush->Color = clBlack;
        Canvas->Brush->Color = clWhite;
```

```
        Canvas->Pen->Color = clBlack;
        Canvas->Pen->Width = 3;
        Canvas->Ellipse(350-300,350-300,350+300,350+300);
        Canvas->Ellipse(350-150,350-150,350+150,350+150);
        Canvas->Brush->Color = clBlack;
        Canvas->Pen->Color = clBlack;
        Canvas->Ellipse(350-5,350-5,350+5,350+5);
        Canvas->Pen->Color = clBlack;
        Canvas->Ellipse(350-7,350-7,350+7,350+7);
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button2Click(TObject *Sender)
{
        //close
        //Form1->Visible=true;
        Form4->Visible=false;
        fclose(fp);
        Form4->Close();
}
//---------------------------------------------------------------------------
//Scanning Bar
void __fastcall TForm4::Timer2Timer(TObject *Sender)
{
// If the current color is red scan to green
    if( shpRed->Brush->Color == clRed)
    {
    // Change the color to green
        while (scanstop == 1)
        {
        scan = 1;
        Timer2->Interval = 2000;
        shpRed->Brush->Color    = clBlack;
        shpYellow->Brush->Color = clBlack;
        shpGreen->Brush->Color  = clGreen;
        shpBlue->Brush->Color   = clBlack;
        while (scanstop == 0)
        {
        Timer2->Interval = 0;
        shpRed->Brush->Color    = clBlack;
        shpYellow->Brush->Color = clBlack;
        shpGreen->Brush->Color  = clGreen;
        shpBlue->Brush->Color   = clBlack;
        break;}
        break;}
        if (activescan == 1)
        {
        scan =1;
        shpRed->Brush->Color    = clBlack;
        shpYellow->Brush->Color = clBlack;
        shpGreen->Brush->Color  = clGreen;
        shpBlue->Brush->Color   = clBlack;
        }
    }
    // But if the color is green scan to yellow
    else if( shpGreen->Brush->Color == clGreen)
    {
```

```
    // Change the color to yellow
    while (scanstop == 1)
    {
    scan=2;
    Timer2->Interval = 2000;
    shpRed->Brush->Color    = clBlack;
    shpYellow->Brush->Color = clYellow;
    shpGreen->Brush->Color  = clBlack;
    shpBlue->Brush->Color   = clBlack;
    while (scanstop == 0)
    {Timer2->Interval = 0;
    shpRed->Brush->Color    = clBlack;
    shpYellow->Brush->Color = clYellow;
    shpGreen->Brush->Color  = clBlack;
    shpBlue->Brush->Color   = clBlack;
    break;}
    break;}
    if (activescan == 2)
    {
    scan = 2;
    shpRed->Brush->Color    = clBlack;
    shpYellow->Brush->Color = clYellow;
    shpGreen->Brush->Color  = clBlack;
    shpBlue->Brush->Color   = clBlack;
    }
}
// But if the color is yellow scan to blue
else if( shpYellow->Brush->Color == clYellow)
{
    // Change the color to Blue
    while (scanstop == 1)
    {
    scan = 3;
    Timer2->Interval = 2000;
    shpRed->Brush->Color    = clBlack;
    shpYellow->Brush->Color = clBlack;
    shpGreen->Brush->Color  = clBlack;
    shpBlue->Brush->Color   = clBlue;
    while (scanstop == 0)
    {Timer2->Interval = 0;
    shpRed->Brush->Color    = clBlack;
    shpYellow->Brush->Color = clBlack;
    shpGreen->Brush->Color  = clBlack;
    shpBlue->Brush->Color   = clBlue;
    break;}
    break;}
    if(activescan == 3)
    {
    scan = 3;
    shpRed->Brush->Color    = clBlack;
    shpYellow->Brush->Color = clBlack;
    shpGreen->Brush->Color  = clBlack;
    shpBlue->Brush->Color   = clBlue;
    }
}
// But if the color is Blue scan to Red
```

```
   else if (shpBlue->Brush->Color   == clBlue)
   {
      while (scanstop == 1)
      {
      scan = 4;
      Timer2->Interval = 2000;
      shpRed->Brush->Color    = clRed;
      shpYellow->Brush->Color = clBlack;
      shpGreen->Brush->Color  = clBlack;
      shpBlue->Brush->Color   = clBlack;
      while (scanstop == 0)
      {Timer2->Interval = 0;
      shpRed->Brush->Color    = clRed;
      shpYellow->Brush->Color = clBlack;
      shpGreen->Brush->Color  = clBlack;
      shpBlue->Brush->Color   = clBlack;
      break;}
      break;}
      if(activescan == 4)
      {
      scan = 4;
      shpRed->Brush->Color    = clRed;
      shpYellow->Brush->Color = clBlack;
      shpGreen->Brush->Color  = clBlack;
      shpBlue->Brush->Color   = clBlack;
      }
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
stamp = stamp + 1;
}
//---------------------------------------------------------------------------
```

APPENDIX D

MATLAB Post Processing Code

D.1 NONLINEAR FILTERING POST PROCESSING CODE – CLEANING

```
%Sara Sibenaller
%Human Engineering Research Laboratories
%Nonlinear Filtering

clear all; clc;
for sn=1:26
%Choose Participant's Data to Parse
if sn<10
    file_name = ['Joystick_data_visit_1_0',num2str(sn),'_clean.txt'];
else
    file_name = ['Joystick_data_visit_1_',num2str(sn),'_clean.txt'];
end
subject=load(file_name);
visit = 1;
dir_d=subject(:,1);
spd_d=subject(:,2);
%rename each trial and target combination
dir_v=0;
spd_v=0;
dir_f=0;
spd_f=0;
for i = 1:(length(subject))
    %Calculate force from digitial speed and direction
    % 3V - 9V => 0 - 4096.
    dir_v(i) = 3/2048*dir_d(i)+3; % do not add parenthese
    spd_v(i) = 3/2048*spd_d(i)+3;
    % 2. convert V to N
    % this function converts the voltage value to newtons
    dir_f(i)=9.4222*dir_v(i)-56.834;
    spd_f(i)=9.4222*spd_v(i)-56.834;
end
    subject(:,14)=dir_f';
    subject(:,15)=spd_f';
    %new(:,16)=f';
for i=1:10
    f=find(subject(:,12)== i, 1, 'last');  %trial
    s=find(subject(:,12)== i, 1, 'first'); %count the number of rows

    for j = s:f
        new(j-(s-1),:) = subject(j,:);     %new = rows 1 to finish
    end
    for k = 0:10                           %attempts
        ff=find(new(:,9) == k, 1, 'last'); %count the number of rows
        ss=find(new(:,9) == k, 1, 'first');
        for m = ss:ff+1
            if k==9
```

```
                m= ss:ff;
            end
            newer(m-(ss-1),:)=new(m,:);
        end
        timer= ((1:length(newer))/100)';   %time based on sampling 100 Hz
        newer(:,13)=timer;
        if sn<10
        if i<10
            attempt=new(ss,9);
            new_file(1,1:13)=['NFs0',int2str(sn),'v0',int2str(visit),...
                't0',int2str(i),'a', int2str(k)];
        elseif i>=10
            attempt=new(ss,9);
            new_file(1,1:13)=['NFs0',int2str(sn),'v0',int2str(visit),...
                't',int2str(i),'a', int2str(k)];
        end
        elseif sn>=10
        if i<10
            attempt=new(ss,9);
            new_file(1,1:13)=['NFs',int2str(sn),'v0',int2str(visit),...
                't0',int2str(i),'a', int2str(k)];
        elseif i>=10
            attempt=new(ss,9);
            new_file(1,1:13)=['NFs',int2str(sn),'v0',int2str(visit),...
                't',int2str(i),'a', int2str(k)];
        end
        end

            new_file=char(new_file);
            new_file
            save (new_file, 'newer')
    clear new_file attempt last ff ss ll l newer;
    end
    clear new trial;
end
clear sn subject dir_d spd_d dir_f spd_f;
end
```

D.2 NONLINEAR FILTERING POST PROCESSING CODE – CALCULATION

```
%Sara Sibenaller
%Human Engineering Research Laboratories
%Quantification of Athetoid Motion
clear, clc;
visit = 1;
%Output Filename for Average Variables-------------------------------------
ofilename=zeros(1,15);
ofilename(1,1:10)=('NFData.xls');
ofilename=char(ofilename);
AveVar = fopen(ofilename, 'a');
fprintf(AveVar, '%7s\t %5s\t %7s\t %9s\t %9s\t %7s\t %12s\t %13s\t %12s\t
%16s\t %7s\t %11s\t %12s\t %13s\t %13s\t %8s\t %8s\t %13s\t %8s\t %8s\t %7s\t
%7s\t %7s\t %7s\t %17s\t %22s\t %17s\t %10s\t %9s\t %9s\t %9s\t %9s\t %7s\t
%13s\t %8s\t %8s\t %13s\t %8s\t %8s\t %7s\t %7s\t %7s\t %7s\t %17s\t %22s\t
%17s\t %10s\t %9s\t %9s\t %9s\t %9s\t %9s\t %9s\t %7s\n',...

'Subject','Trial','Attempt','TotalTime','IdealDist','TotDist','IndexDiff','Ta
skPrecision','IndexPerform','PercentDistError','RxnTime','NumberPause','MeanP
auseDur','NumAccelPeaks','MeanPeakAccel','AveVeloc','MaxVeloc','TimeBeforeTE'
,'TimeAfterTE','AveAccel','MaxAccel','AveJerk','MaxJerk','AveSnap','MaxSnap',
'num_large_submove','mean_dur_large_submove','ave_submove_veloc',
'sub_before_enter', 'sub_after_enter', 'percent_in_target',
'number_slipoffs', 'status','NTT', 'NTD', 'NPDE', 'NNP', 'NMPD', 'NNA',
'NMPA', 'NAV', 'NTBTE', 'NTATE', 'NAA', 'NNS', 'NMDS', 'NASV', 'NSBE',
'NSAE', 'NPI', 'NNSLIP', 'NSTAT','CHECK','DSTime');
for sn = 1:26   %subject number
    sn
    if sn==14
        continue
    end
%Name and Load 100 unique trials per participant--------------------------
%trials 1 through 10
for t=1:10
        filename=zeros(1,17);
        pxyplotname=zeros(1,17);
        pxyfilename=zeros(1,20);
        pxyfilename2=zeros(1,20);
    %Adjust for Subject Number with 1 or 2 digits
    if sn<10
        filename(1,1:5)=['NFs0',int2str(sn)];
        pxyplotname(1,1:5)=['NFS0',int2str(sn)];
        pxyfilename(1,1:8)=['NFpxyS0',int2str(sn)];
        pxyfilename2(1,1:8)=['NFpxyS0',int2str(sn)];
    else
        filename(1,1:5)=['NFs',int2str(sn)];
        pxyplotname(1,1:5)=['NFS',int2str(sn)];
        pxyfilename(1,1:8)=['NFpxyS',int2str(sn)];
        pxyfilename2(1,1:8)=['NFpxyS',int2str(sn)];
    end
    %Adjust for Trial Number with 1 or 2 digits
    if t<10
        filename(1,6:12)=['v0',int2str(visit),'t0',int2str(t),'a'];
        pxyplotname(1,6:12)=['V0',int2str(visit),'T0',int2str(t),'A'];
        pxyfilename(1,9:15)=['V0',int2str(visit),'T0',int2str(t),'A'];
```

```
        pxyfilename2(1,9:15)=['V0',int2str(visit),'T0',int2str(t),'A'];
    else
        filename(1,6:12)=['v0',int2str(visit),'t',int2str(t), 'a'];
        pxyplotname(1,6:12)=['V0',int2str(visit),'T',int2str(t), 'A'];
        pxyfilename(1,9:15)=['V0',int2str(visit),'T',int2str(t),'A'];
        pxyfilename2(1,9:15)=['V0',int2str(visit),'T',int2str(t),'A'];
    end
    %attempt 1 through 9
    for a=0:9
        filename(1,13:17)=[int2str(a),'.mat'];
        pxyplotname(1,17)=int2str(a);
        pxyfilename(1,16:20)=[int2str(a),'.jpg'];
        pxyfilename2(1,16:20)=[int2str(a),'.fig'];
        filename=char(filename);
        pyxplotname=char(pxyplotname);
        pxyfilename=char(pxyfilename);
        pxyfilename2=char(pxyfilename2);
        load(filename);
        filename;
%LOAD_PARAMETERS=============================================================
direction = newer(:,1);
speed = newer(:,2);
x = newer(:,3);
y = newer(:,4);
xstart=x(1,1);
ystart=y(1,1);
xtarget = newer(:,5);
ytarget = newer(:,6);
xfinish=xtarget(1,1);
yfinish=ytarget(1,1);
pxtarget = newer(:,7);
pytarget = newer(:,8);
total = newer(:,9);
miss = newer(:,10);
hit = newer(:,11);
trial_number = newer(:,12);
timer = newer(:,13);
%Actual target width based on subject
if sn == 1
    width = 48;
else
    width = 96;
end
%Downsample_to_20_Hz from 100 Hz
for i=1:(length(newer))
    xy(i)=sqrt(x(i)^2 + y(i)^2);
end
xy_ds = downsample(xy,5);
x_ds = downsample(x,5);
y_ds = downsample(y,5);
time_ds = downsample(timer,5);
%BASIC_VARIABLES=============================================================
%ideal path distance (pixels)
ideal_dist = sqrt((xfinish-xstart)^2+(yfinish-ystart)^2);
%actual distance traveled
dist = length(newer)-1;
for i = 1:(length(newer))-1;
```

```
    dist(i) = sqrt((x(i+1)-x(i)).^2 + (y(i+1)-y(i)).^2);  %distance formula
end
tot_dist = sum(dist);
%Time to acquire target
tot_time_place = find(timer, 1, 'last');
tot_time = timer(tot_time_place,1);
%FITTS_LAW_PERFORMANCE_VARIABLES=========================================
%Index of Difficulty
%calculation from ISO standards document
ID = log2((ideal_dist*width)/width); %Calculation using ISO
%Task Precision
if ID <=4
    tp=1;
elseif ID>6
    tp=3;
else
    tp=2;
end
%Index of Performance
IP=ID/tot_time;
%Percentage Distance Error
PDE=((abs(tot_dist-ideal_dist))/tot_dist)*100;
%KINEMATIC_VARIABLES=====================================================
%Velocity (pixels per second)
veloc=zeros(1,(length(xy_ds)));
for i=1:(length(xy_ds)-1);
    if i==1
        veloc(i)=0;
    else
        veloc(i)= sqrt((x_ds(i)-x_ds(i-1))*(x_ds(i)-x_ds(i-1))...
            +(y_ds(i)-y_ds(i-1))*(y_ds(i)-y_ds(i-1)))/.05;
    end
end

%veloc= abs(veloc);
ave_veloc = mean(veloc);
max_veloc = max(veloc);
MaxVelIndex = find(veloc==max_veloc);
mode_veloc = mode(veloc);
%Acceleration (pixels per second^2)
accel=zeros(1,(length(xy_ds)));
for i=1:(length(xy_ds)-2)
    if i==1
        accel(i)=0;
    else
        accel(i)=(veloc(i)-veloc(i-1))/(.05);
    end
end
ave_accel = mean(accel);
max_accel = max (accel);
mode_accel = mode(accel);
%Number of Acceleration / Deceleration Impulses
accelzero = find(diff(sign(accel)));
accelmax = find(sign(accel(2:end-1)-accel(1:end-2))+sign(accel(2:end-1)-...
    accel(3:end))>0) +1;
accelmin = find(sign(accel(2:end-1)-accel(1:end-2))+sign(accel(2:end-1)-...
    accel(3:end))<0) +1;
```

```
max_accel = max(accel);
min_accel = min(accel);
if length(accelmax) < 1
    max_accel=0;
    accelmax_w0=0;
    accelmax_n0=0;
    abs_mean_peak_max_accel=0;
    number_max_accel=0;
else
for i = 1:length(accelmax)
if (accel(accelmax(i)) >= (max_accel*0.05))
    accelmax_w0(i)= accelmax(i);
end
end
accelmax_n0 = nonzeros(accelmax_w0);
abs_mean_peak_max_accel = abs(mean(accelmax_n0)); %mean peak accel
number_max_accel = length(accelmax_w0); %number of accel peaks
end
if length(accelmin) < 1
    min_accel=0;
    accelmin_w0=0;
    accelmin_n0=0;
    abs_mean_peak_min_accel=0;
    number_min_accel=0;
else
for i = 1:length(accelmin)
if (accel(accelmin(i)) <= (min_accel*0.05))
    accelmin_w0(i)= accelmin(i);
end
end
accelmin_n0 = nonzeros(accelmin_w0);
abs_mean_peak_min_accel = abs(mean(accelmin_n0)); %mean peak accel
number_min_accel = length(accelmin_w0); %number of accel peak
end
mean_peak_accel = (abs_mean_peak_max_accel+abs_mean_peak_min_accel)/2;
number_accel = number_min_accel + number_max_accel;
%Jerk (pixels per second^3)
jerk=zeros(1,(length(xy_ds)));
for i=1:(length(xy_ds)-3)
    if i==1
        jerk(i)=0;
    else
        jerk(i)= (accel(i)-accel(i-1))/(.05);
    end
end
ave_jerk = mean(jerk);
max_jerk = max (jerk);
mode_jerk = mode(jerk);
%Jerk Zero Crossing (min)
jerkzero = find(diff(sign(jerk)));
%Snap (pixels per second^4)
snap=zeros(1,(length(xy_ds)));
for i=1:(length(xy_ds)-4)
    if i==1
        snap(i)=0;
    else
        snap(i)=(jerk(i)-jerk(i-1))/(.05);
```

```
    end
end
ave_snap=mean(snap);
max_snap=max(snap);
mode_snap=mode(snap);
%PAUSESandSUBMOVEMENTS=================================================
%PAUSES
%Identify Pauses Using Velocity
for i=1:(length(veloc));
    if veloc(i) == 0;
        if i ==1 && veloc (i+1)>0 && veloc(i+2)==0
            pause = 0;
        else
    pause(i) = 1;
        end
    else
    pause(i) = 0;
    end
end
%assign indicies to pause moments
pause_ind=find(pause==1)';
pe=pause_ind(length(pause_ind),1);
pf=pause_ind(1,1);
one_pause = (pe-pf)+1;
p=0;
pr=1;
if (one_pause==length(pause_ind))
    p=length(pause_ind);
    pause_dur(pr,1)=p;
    p=0;
    pr=pr+1;
else
for i = 1:((length(pause_ind)))
    if i==(length(pause_ind))
        p=p+1;

    elseif((pause_ind(i+1))-(pause_ind(i))==1)
        p=p+1;
    elseif ((pause_ind(i+1))-(pause_ind(i))>1)
        pause_dur(pr,1)=p;
        p=0;
        pr=pr+1;
    end
   if (i == length(pause_ind))
        pause_dur(pr,1)=p;
        p=0;
        pr=pr+1;
   end
end
end
%number of pauses not including reaction time pause
if pause(1,1) == 1;
number_pause = (length(pause_dur)-1);
pause_dur_noRXN = pause_dur(2:end,1);
else
number_pause = length(pause_dur)
pause_dur_noRXN = pause_dur
```

```
end
%mean pause duration
if number_pause == 0
    mean_pause_dur = 0;
else
mean_pause_dur = (mean((pause_dur_noRXN)))*.05;
end
%Reaction Time
if (pause(1,1)==0)
    react_time = 0;
else
    react_time = (pause_dur(1,1))*.05;
end
%identify submovements using pauses
submove_ind = find(pause==0)';
if size(submove_ind) == [0,1];
    se = 0;
    sf = 0;
else
se=submove_ind(length(submove_ind),1);
sf=submove_ind(1,1);
one_submove = (se-sf)+1;
s=0;
sr=1;
end
if (one_submove==length(submove_ind))
    s=length(submove_ind);
    submove_dur(sr,1)=s;
    s=0;
    sr=sr+1;
else
for i = 1:((length(submove_ind)))
    if i==(length(submove_ind))
        s=s+1;

    elseif((submove_ind(i+1))-(submove_ind(i))==1)
        s=s+1;
    elseif ((submove_ind(i+1))-(submove_ind(i))>1)
        submove_dur(sr,1)=s+2;
        s=0;
        sr=sr+1;
    end
   if (i == length(submove_ind))
        submove_dur(sr,1)=s+2;
        s=0;
        sr=sr+1;
   end
end
end
%number of submovements and submovement duration
if size(submove_ind) == [0,1];
    number_submove = 0;
    mean_submove_dur = 0;
    submove_ind=0;
else
%number of submoves not including reaction time submove
number_submove = length(submove_dur);
```

```
%mean submove duration
if number_submove == 0
    mean_submove_dur = 0;
else
mean_submove_dur = (mean((submove_dur)))*.05;
end
end
%Submovement velocity
r3=1;
for i=1:2:(length(submove_ind)-1)

    sf=i;
    se=i+1;
    sfi=submove_ind(sf,1);
    sei=submove_ind(se,1);
    for j=sfi:sei
        sub_veloc(j-(sfi-1),1)=veloc(j);
    end
    mean_sub_veloc(r3)=mean(sub_veloc);
    r3=r3+1;
end
average_submove_veloc = mean(mean_sub_veloc);
clear sub_veloc;

%TARGET ENTRY=================================================================
%Accuracy_Variables-----------------------------------------------------------
tr=1;
for i=1:length(x_ds)
if ((x_ds(i)-xfinish)*(x_ds(i)-xfinish)+(y_ds(i)-yfinish)*...
        (y_ds(i)-yfinish) <= width^2)
    intarget(tr,1)=1;
    tr=tr+1;
else
    intarget(tr,1)=0;
    tr=tr+1;
end
end
intarget_ind = find(intarget == 1);
outtarget_ind = find(intarget ==0);
%Percent In/Out Target
length_in_target = length(intarget_ind);
length_out_target = length(outtarget_ind);
length_total_trial = length(x_ds);
percent_in = (length_in_target/length_total_trial)*100;
percent_out = (length_out_target/length_total_trial)*100;
%Enter/Exit Indicies
ee=1;
for i=1:length(intarget)
    if i==1
        enterexit(ee,1)=0;
        ee=ee+1;
    elseif intarget(i) == 1
        if intarget(i-1)==0
            enterexit(ee,1)=1;
            ee=ee+1;
        else
```

```
            enterexit(ee,1)=0;
            ee=ee+1;
        end
    elseif intarget(ee,1)==0;
        if intarget(i-1)==1
            enterexit(ee,1)=2;
            ee=ee+1;
        else
            enterexit(ee,1)=0;
            ee=ee+1;
        end
    end
end
%Slip Offs
slipoff_ind = find(enterexit==2);
number_slipoffs = length(slipoff_ind);
%Number ReEntries
enter_ind = find(enterexit==1);
number_entries = length(enter_ind);
%First Target Entry Indicy
if size(enter_ind)==[0,1];
    first_entry_ind =0;
else
first_entry_ind = enter_ind(1,1);
end
sbe=0;
sae=0;
total_sub=0;
if submove_ind(1,1)== 0;
    sbe = 0;
    sae = 0;
    total_sub = 0;
else
for i=1:2:length(submove_ind)
if first_entry_ind ==0
    sbe=0;
elseif submove_ind(i) <= first_entry_ind
    sbe=sbe+1;
end
    total_sub = total_sub+1;
end
sae = total_sub - sbe;
end
%Time Before and After First Target Entry
if first_entry_ind ==0
    TimeBeforeTE = 0;
else
    if (length(time_ds))< (first_entry_ind)
        TimeBeforeTE = 0;
    else
    TimeBeforeTE = time_ds(first_entry_ind) - time_ds(1);
    end
end
TimeAfterTE = tot_time - TimeBeforeTE;
%Acquire Target?
last_hit_place = find(hit,1,'last');
last_hit = hit(last_hit_place,1);
```

```
second2last_hit_place = last_hit_place - 1;
second2last_hit = hit(second2last_hit_place,1);
if last_hit > second2last_hit
    status = 1;
else
    status = 0;
end
%Adjust Total Time
if tot_time == 19.99 && status == 0
    tot_time = 20.00;
end
%Normalize Variables
NTT=tot_time/ideal_dist;
NTD=tot_dist/ideal_dist;
NPDE=PDE/ideal_dist;
NNP=number_pause/ideal_dist;
NMPD=mean_pause_dur/ideal_dist;
NNA=number_accel/ideal_dist;
NMPA=mean_peak_accel/ideal_dist;
NAV=ave_veloc/ideal_dist;
PTBTE=TimeBeforeTE/tot_time;
PTATE=TimeAfterTE/tot_time;
NAA=ave_accel/ideal_dist;
NNS=number_submove/ideal_dist;
NMDS=mean_submove_dur/ideal_dist;
NASV=average_submove_veloc/ideal_dist;
NSBE=(sbe/total_sub);
NSAE=(sae/total_sub);
NPI=percent_in/ideal_dist;
NNSLIP=number_slipoffs/ideal_dist;
NSTAT=status/ideal_dist;
subpausecheck = (number_submove*mean_submove_dur)+...
    (number_pause*mean_pause_dur)+ react_time;
DStimeplace = find(time_ds,1,'last');
finalDStime = time_ds(DStimeplace, 1);

%output to excel file
AveVar = fopen(ofilename, 'a');
fprintf(AveVar, '%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\n ', sn , t, a,
tot_time, ideal_dist, tot_dist, ID, tp, IP, PDE, react_time, number_pause,
mean_pause_dur, number_accel, mean_peak_accel, ave_veloc, max_veloc,
TimeBeforeTE, TimeAfterTE, ave_accel, max_accel, ave_jerk, max_jerk,
ave_snap, max_snap, number_submove, mean_submove_dur, average_submove_veloc,
sbe, sae, percent_in, number_slipoffs, status, NTT, NTD, NPDE, NNP, NMPD,
NNA, NMPA, NAV, PTBTE, PTATE, NAA, NNS, NMDS, NASV, NSBE, NSAE, NPI, NNSLIP,
NSTAT, subpausecheck, finalDStime);
clear s se sf enter_ind xy xy_ds accel sbe total_sub sae pbe pae
number_slipoffs percent_in percent_out length_in_target length_out_target
accelzero accelmax tp accelmin max_accel min_accel accelmin_n0 accelmin_w0
accelmax_n0 accelmax_w0 mean_peak_min_accel K AveKn0 dist tot_dist ideal_dist
IP ID PI n_ave_veloc n_ave_accel n_ave_jerk n_ave_snap n_number_accel
n_abs_mean_peak_accel VelMaxBA VelMaxTimeBATargetEntry PE c rxn_time pin
percentin react_time number_pause mean_pause_dur number_large_submove
mean_dur_large_submove average_submove_veloc se sei sf sfi j sub_veloc
```

```
submove_ind react_time submove pause pause_ind pause pause_dur move move_ind
move_dur VelMaxTimeFromTargetEntry TimeBeforeTE TimeBeforeTE intarget NTT NTD
NPDE NNP NMPD NNA NMPA NAV NTBTE NTATE NAA NNS NMDS NASV NSBE NSAE NPI NNSLIP
NSTAT subpausecheck meanDStime submove_dur submove_ind;

end
end
fclose('all')
end
```

D.3 NONLINEAR FILTERING POST PROCESSING CODE – AVERAGING

```
%Sara Sibenaller
%Human Engineering Research Laboratories
%Average variables for each participant

close all, clear, clc;
%load file
data = load('NFdata.txt');
%load output file2
ofilename(1,1:9)=('NFsub.xls');
ofilename=char(ofilename2);
ofilename;
SNVar = fopen(ofilename, 'a');
%--------------------------------------------------------------------------
sn = data (:,1);
t = data (:,2);
a = data (:,3);
for sn = 1:26
    lastsn=find(data(:,1) == sn, 1, 'last');
    firstsn=find(data(:,1) == sn, 1, 'first');
    for blocksn = firstsn:lastsn
        datasn((blocksn-(firstsn-1)),:)=data(blocksn,:);
    end
        row1 = [datasn(1,1)];
        row2 = mean(datasn(:,4:7));
        row3 = median(datasn(:,8));
        row4 = mean(datasn(:,9:17));
        row5 = mean(datasn(:,18));
        row6 = mean(datasn(:,19:29));
        row7 = sum(datasn(:,30));
        row8 = mean(datasn(:,31:54));
        fprintf(SNVar, '%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t%f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\n',
row1,row2,row3,row4,row5,row6,row7,row8);
    end
    clear datat data datasn;
fclose('all')
```

D.4 CUSTOMIZED CONTROL POST PROCESSING CODE – CLEANING

```
%Sara Sibenaller
%Human Engineering Research Laboratories
%Customized Control

clear all, clc;
%Choose DATA to Cut
[filename,pathname]=uigetfile('*.*','Select a Subject File');
new=load([pathname,filename]);
sn=input('Which subject did you load?');
visit=input(['Which visit for subject ' num2str(sn) ' did you load?']);
set=input(['Which set did you load?']); %set1 = switch, set2=noswitch
dir_d=new(:,1);
spd_d=new(:,2);
%rename each trial and target combination -------------------------------
for i = 1:(length(new))
        %Calculate force from digitial speed and direction variables
        %1. convert digital to voltage 3V - 9V => 0 - 4096.
        dir_v(i) = 3/2048*dir_d(i)+3; % do not add parenthese
        spd_v(i) = 3/2048*spd_d(i)+3;
        % 2. convert voltage value to newtons
        dir_f(i)=9.4222*dir_v(i)-56.834;
        spd_f(i)=9.4222*spd_v(i)-56.834;
        f = sqrt(dir_f.^2 + spd_f.^2);
end
        new(:,12)=dir_f';
        new(:,13)=spd_f';
        new(:,14)=f';
    %Separate data into 32 attempts
for k = 0:31
    %count the number of elements of new = to k ->last
   ll=find(new(:,7) == k, 1, 'last');
        ff=find(new(:,7) == k, 1, 'first');
        %newer = rows 1 to last of k in new
        for m = ff:ll+1
            newer(m-(ff-1),:)=new(m,:);
        end
        %Create timer (data sampled at 100 Hz)
        timer= ((1:length(newer))/100)';
        newer(:,15)=timer;
        % Nomenclature (CCS0_V0_S0_A_ or CCS0_V0_S0_A_) for naming .txt files
        % accoding to subject, visit, set, and attempt
        new_file(1,1:2)='CC';
        if sn>=10
            new_file(1,3:5)=['S',int2str(sn)];
        else
            new_file(1,3:5)=['S0',int2str(sn)];
        end
        if k<10
            new_file(1,6:14)=['V0',int2str(visit), 'S0',...
                int2str(set), 'A0',int2str(k)];
        else
            new_file(1,6:14)=['V0',int2str(visit), 'S0',...
                int2str(set), 'A',int2str(k)];
        end
```

```
            new_file=[pathname, char(new_file)];
            disp('Creating new file:');
            new_file
            save (new_file, 'newer')
            clear new_file attempt last ff ll m newer;
end
```

D.5 CUSTOMIZED CONTROL POST PROCESSING CODE –CALCULATION

```
%Sara Sibenaller
%Human Engineering Research Laboratories
%Customized Control

close all, clear, clc;
ofilename(1,1:10)=('CCdata.xls');
ofilename=char(ofilename);
ofilename
Var = fopen(ofilename, 'a');
%--------------------------------------------------------------------------
%Run through 32 trials of data
for sn = 1:11
    if sn<10
    filename(1,1:5)=['CCS0',int2str(sn)];
    else
    filename(1,1:5)=['CCS',int2str(sn)];
    end

    for visit = 1:2
        %exclude missing data from analysis
        for s=1:2
            if visit ==1 & s==2
                continue
            end
            if sn == 4 & visit ==1 & s==1
                continue
            end
            if sn == 6 & visit ==2
                continue
            end
            if sn == 7
                if visit==1 & s==1
                    continue
                end
                if visit==2 & s==1
                    continue
                end
            end
            if sn == 9 & visit ==2
                continue
            end
for a=0:31
    if a<10

filename(1,6:18)=['V0',int2str(visit),'S0',int2str(s),'A0',int2str(a),'.mat'
];
    else

filename(1,6:18)=['V0',int2str(visit),'S0',int2str(s),'A',int2str(a),'.mat'];
    end
    filename=char(filename);
    load(filename); %Assign the file to the variable newer
    filename
```

```
%-------------------------------------------------------------------------
%load parameters
direction = newer(:,1);
speed = newer(:,2);
x = newer(:,3);
y = newer(:,4);
xtarget = newer(:,5);
ytarget = newer(:,6);
total = newer(:,7);
miss = newer(:,8);
hit = newer(:,9);
target_number=newer(:,10);
forcex=newer(:,12);
forcey=newer(:,13);
forcemag=newer(:,14);
timer = newer(:,15);
xfinish=xtarget(1,1);
yfinish=ytarget(1,1);
xstart=x(1,1);
ystart=y(1,1);
%Time to acquire target
tot_time_place = find(timer, 1, 'last');
tot_time = timer(tot_time_place,1);
%Target number
t_num_place = find(target_number,1,'first');
t_num=target_number(t_num_place,1);
%hit or miss?
last_hit_place = find(hit,1,'last');
last_hit = hit(last_hit_place,1);
second2last_hit_place = last_hit_place - 1;
second2last_hit = hit(second2last_hit_place,1);
if last_hit > second2last_hit
    %status = 'hit';
    status = 1;
else
    %status = 'miss';
    status = 0;
end
%Force
AveForce = mean(forcemag);
MaxForce = max(forcemag);
ModeForce = mode(forcemag);
width = 96;
%BASIC_VARIABLES=============================================================
%ideal path distance (pixels)
ideal_dist = sqrt((xfinish-xstart)^2+(yfinish-ystart)^2);
%actual distance traveled
dist = length(newer)-1;
for i = 1:(length(newer))-1;
    dist(i) = sqrt((x(i+1)-x(i)).^2 + (y(i+1)-y(i)).^2);  %distance formula
end
tot_dist = sum(dist);
%ID calculation from ISO standards document
ID = log2((ideal_dist*width)/width); %Calculation using ISO
%EID = log2((tot_dist*eff_width)/eff_width)
%Task Precision
if ID <=4
```

```
    tp=1;
elseif ID>6
    tp=3;
else
    tp=2;
end
%Index of Performance
IP=ID/tot_time;
%Percentage Distance Error (CHECK)
PDE= ((abs(tot_dist-ideal_dist))/tot_dist);
%Downsample_to_20_Hz from 100 Hz
for i=1:(length(newer))
    xy(i)=sqrt(x(i)^2 + y(i)^2);
end
xy_ds = downsample(xy,5);
time_ds = downsample(timer,5);

veloc=zeros(1,(length(xy_ds)));
for i=1:(length(xy_ds)-1);
    veloc(i)= sqrt((x_ds(i)-x_ds(i-1))*(x_ds(i)-x_ds(i-1))...
            +(y_ds(i)-y_ds(i-1))*(y_ds(i)-y_ds(i-1)))/.05;
end
%Identify Pauses Using Velocity
for i=1:(length(veloc));
    if veloc(i) == 0;
    pause(i) = 1;
    else
    pause(i) = 0;
    end
end
%assign indicies to pause moments (WILL USE LATER FOR PLOTTING PAUSES)
pause_ind=find(pause==1)';
pe=pause_ind(length(pause_ind),1);
pf=pause_ind(1,1);
one_pause = (pe-pf)+1;
p=0;
pr=1;
if (one_pause==length(pause_ind))
    p=length(pause_ind);
    pause_dur(pr,1)=p;
    p=0;
    pr=pr+1;
else
for i = 1:((length(pause_ind)))
    if i==(length(pause_ind))
        p=p+1;
    elseif((pause_ind(i+1))-(pause_ind(i))==1)
        p=p+1;
    elseif ((pause_ind(i+1))-(pause_ind(i))>1)
        pause_dur(pr,1)=p;
        p=0;
        pr=pr+1;
    end
end
end
%Reaction Time
if (pause(1,1)==0)
```

```
    react_time = 0;
else
    react_time = (pause_dur(1,1))*.05;
end
%Var = fopen(ofilename, 'a');
fprintf(Var, '%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\n ', sn, visit, s, a, status, ID, IP, PDE, tot_time, react_time, AveForce,
MaxForce, ModeForce);
clear xy xy_ds veloc pause pause_ind pause pause_dur react_time;
            end
        end
    end
end
fclose('all')
```

D.6 CUSTOMIZED CONTROL POST PROCESSING CODE – AVERAGING

```
%Sara Sibenaller
%Human Engineering Research Laboratories
%Customized Control Computer Access Data
%Averaged Trials

close all, clear, clc;
%load file
data = load('CCdata.xls')
%load output file
ofilename(1,1:13)=('CCavedata.xls');
ofilename=char(ofilename);
ofilename
AveVar = fopen(ofilename, 'a');
%-------------------------------------------------------------------------
sn = data (:,1);
visit = data (:,2);
s = data (:,3);
a = data (:,4);
%Run through 32 trials of data
for sn = 1:11
    lastsn=find(data(:,1) == sn, 1, 'last');
    firstsn=find(data(:,1) == sn, 1, 'first');
    for blocksn = firstsn:lastsn
        datasn((blocksn-(firstsn-1)),:)=data(blocksn,:);
    end
    for visit = 1:2
        lastvisit=find(datasn(:,2) == visit, 1, 'last');
        firstvisit=find(datasn(:,2) == visit, 1, 'first');
        for blockvisit = firstvisit:lastvisit
            datavisit((blockvisit-(firstvisit-1)),:)=datasn(blockvisit,:);
        end
        for s=1:2
            %exclude missing data
            if visit ==1 & s==2
                continue
            end
            if sn == 4 & visit ==1 & s==1
                continue
            end
            if sn == 6 & visit ==2
                continue
            end
            if sn == 7
                if visit==1 & s==1
                    continue
                end
                if visit==2 & s==1
                    continue
                end
            end
            if sn == 9 & visit ==2
                continue
            end
            lastset=find(datavisit(:,3) == s, 1, 'last');
```

```
            firstset=find(datavisit(:,3) == s, 1, 'first');
            for blockset = firstset:lastset
                dataset((blockset-(firstset-1)),:)=datavisit(blockset,:);
            end
            for a=0:31
             status = dataset (:,5);
             ID = dataset(:,6);
             IP = dataset(:,7);
             PDE = dataset(:,8);
             tot_time = dataset(:,9);
             react_time = dataset(:,10);
             AveForce = dataset(:,7);
             MaxForce = dataset(:,8);
             ModeForce = dataset(:,9);

             percenthit = ((sum (status))/32)*100;
            AveID = mean(ID);
            AveIP = mean(IP);
            AveRxnTime = mean(react_time);
            AvePDE = mean(PDE);
            AveTime = mean(tot_time);
            AveForce = mean(AveForce);
            AveMaxForce = mean(MaxForce);
            AveModeForce = mean(ModeForce);
            end
            fprintf(AveVar, '%f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\t %f\t %f\n ', sn, visit, s, percenthit, AveTime, AveRxnTime, AveID,
AveIP, AvePDE, AveForce, AveMaxForce, AveModeForce);
            clear status tot_time AveForce MaxForce ModeForce percenthit
AveTime AveForce AveMaxForce AveModeForce ID IP PDE react_time AveRxnTime
AveID AveIP AvePDE
        end
    end
    clear dataset datavisit datasn
end
fclose('all')
```

BIBLIOGRAPHY

1. Cerebral Palsy Magazine 2008 [cited 2008 5/25/08].

2. Morris. Definition and Classification of Cerebral Palsy: A Historical Perspective. Dev Med Child Neurol Suppl 2007;109:3-7.

3. Singer H. Movement Disorders in Children. In: Jankovic J, Tolosa E, editors. Parkinson's Disease & Movement Disorders Philedelphia, PA: Lippincott Williams & Wilkins; 2007. p 481-503.

4. Bax M, Goldstein M, Rosenbaum R, Leviton A, Paneth A. Proposed definition and classification of cerebral palsy. United Cerebral Palsy Research & Education Foundation 2005 April 2005.

5. Albanese A, Bentivoglio AR. Botulinum Toxin in Movement Disorders. In: Jankovic J, Tolosa E, editors. Parkinson's Disease and Movement Disorders. Philedelphia, PA: Lippincott Williams and Wilkins; 2007. p 605-19.

6. Penn R. Intrathecal baclofen for severe spasticity. Ann NY Acad Sci 1988;153:157-66.

7. Bohannon R, Smith M. Interrater reliability of a modified Ashworth scale of muscle spasticity. Phys Ther 1987;67(2):206-7.

8. Tarsy D, Simon D. Dystonia. The New England Journal of Medicine 2006;355(8):818-29.

9. Comella C, Leurgans S, Wuu J, Stebbins G, Chmura T. Rating scales for dystonia: a multicenter assessment Movement Disorders 2003;18(3):303-12.

10. Koven LJ, Lamm SS, Koven LJ, Lamm SS. The athetoid syndrome in cerebral palsy. II. Clinical aspects. Pediatrics 1954;14(3):181-92.

11. Busby G. Technology for the disabled and why it matters to you. IEE Colloquium Digest Computers in the service of mankind: Helping the disabled 1997;97(17):1-7.

12. Morris, Jankelowitz, Fung, Clouston, Hayes, Grattan-Smith. Athetosis I: Historical Considerations. Movement Disorders 2002;17(6):1278-80.

13. Spitz M, Machado A, Carvalho R, Maia F, Haddad M, Calegaro D et al. Pseudoathetosis: Report of Three Patients. Movement Disorders 2006;21(9):1520-2.

14. Wilson K. Disorders of Motility and of Muscle Tone. Royal College of the Physicians of London. 1925.

15. Bucy P. The Neural Mechanisms of Athetosis and Tremor. Journal of Neuropathy 1942;1:224-39.

16. Meeteren J, Roebroeck M, Celen E, Donkervoort M, Stam H. Functional activities of the upper extremity of young adults with cerebral palsy: A limiting factor for partitipation? Disability and Rehabilitation 2008;30(5):387-95.

17. Cook A, Hussey S. Control Interfaces for Assistive Technology. In: Conklin K, editor. Assistive Technologies: Principles and Practice. Second ed. St. Louis: Mosby, Inc. ; 2002. p 212-54.

18. Seelman K, Hackett S, Parmanto B, Simpson R, Panchang S. Telecommunications, Computers, and Web Accessibility. In: Cooper RA, Ohnabe H, Hobson DA, editors. An Introduction to Rehabilitation Engineering. Boca Raton, FL: Taylor & Francis Group, LLC; 2007.

19. Logitech. Logitech Trackball Mouse 2008 [cited 2008 7/18]. Available from: URL: www.logitech.com.

20. Penny & Giles Computer Interface 2008 [cited 2008 7/18]. Available from: URL: www.pennyandgiles.com.

21. Forbes Rehab Services Head Pointer 2008 [cited 2008 7/31]. Available from: URL: www.frs-solutions.com.

22. Koester HH, Lopresti E, Ashlock G, McMillan W, Moore P, Simpson R. COMPASS: Software for Computer Skills Assessment. Technology and Persons with Disabilities Conference. Los Angeles, CA; 2007.

23. ISO9241-9. Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9: Requirements for non-keyboard input devices. 2000 2-15-2000.

24. Man DW, Wong MS. Evaluation of computer-access solutions for students with quadriplegic athetoid cerebral palsy. American Journal of Occupational Therapy 2007;61(3):355-64.

25. Cooper R, Jones D, Fitzgerald S, Boninger M, Albright S. Analysis of position and isometric joysticks for powered wheelchair driving. IEEE Trans Biomed Eng 2000;47(7):902-10.

26. Cooper R, Spaeth D, Jones D, Boninger M, Fitzgerald S, Guo S. Comparison of virtual and real electric powered wheelchair driving using a position sensing joystick and an isometric joystick. Medical engineering & physics 2002;24(10):703-8.

27. Spaeth DM. Evaluation of an isometric joystick with control enhancing algorithms for improved driving of electric powered wheelchairs. University of Pittsburgh; 2002.

28. Kimmich C, Sibenaller S, Dicianno BE. Joystick Use for Virtual Power Wheelchair Driving in Individuals with Tremor. RESNA Annual Conference. Washington, DC 2008.

29. Dicianno B, Spaeth D, Cooper R, Fitzgerald S, Boninger M, Brown K. Force Control Strategies While Driving Electric Powered Wheelchairs With Isometric and Movement Sensing Joysticks. IEEE Transactions on Neural Systems and Rehabilitation Engineering 2007;15(1):144-50.

30. Stewart H, Noble G, Seeger BR. Isometric Joystick: A study of Control by Adolescents and Young Adults with Cerebral Palsy. The Australian Occupational Therapy Journal 1991;39:33-9.

31. Rao RS, Seliktar R, Rahman T. Evaluation of an isometric and a position joystick in a target acquisition task for individuals with cerebral palsy. IEEE Trans Rehabil Eng 2000;8(1):118-25.

32. Kaye S. Computer and Internet Use Among People with Disabilities. Disability Statistics Center Institute for Heath and Aging; 2000.

33. MacKenzie S, Kauppinen T, Silfverberg M. Accuracy Measures for Evaluating Computer Input Devices. Proceedings of the SIGCHI conference on Human factors in computing systems; Seattle Washington; 2001.

34. Keates S, Clarkson J, Robinson P. Understanding how to improve the accessibility of computers through cursor control studies. Association for Computing Machinery 2008(3/11/08).

35. Hwang, Keates, Langdon, Clarkson. Mouse Movements of Motion-Impaired Users: A Submovement Analysis ACM ASSETS Conference. Atlanta, GA; 2004.

36. Radwin R, Vanderheiden G, Lin M. A method for evaluating head-controlled computer input devices using Fitts' law. Human Factors 1990;32(4):423-38.

37. Sibenaller S, Ding D, Dicianno B, Cooper R, Riviere C. Kinematic Characteristics of Athetoid Movement During Target Acquisition. Proceedings of the RESNA Annual Conference; Arlington, VA; 2008.

38. Lopez J, Sibenaller S, Ding D, Riviere C. Toward filtering of athetoid motion with neural networks. IEEE Eng Med Biol Soc. Valladolid, Spain; 2007.

39. Olds KC, Sibenaller S, Cooper RA, Ding D, Riviere C. Target Prediction for Icon Clicking by Athetoid Persons. IEEE Int Conf Robot Autom (ICRA). Pasadena, California; 2007.

40. Niku S, Henderson JM. Toward quantification of athetotic movements by frequency spectrum analysis. Journal of Biomechanics 1985;18(1):71-6.

41. Borland C++ Builder. Version 5. Scotts Valley, CA 95066: Inprise Corporation 2000.

42. MATLAB. Version 7.1.0.246 (R14) Service Pack 3. 24 Prime Park Way, Natick, MA 01760-1500; 2005.

43. Sibenaller S, Ding D, Dicianno B, Cooper R. Development of Customizable Algorithms for an Isometric Joystick for Individuals with Dystonia and Chorea RESNA Annual Conference. Phoenix, AZ; 2007.

44. SPSS-Inc. 233 S Wacker Dr, 11th fl, Chicago, IL 60606 2005;14.0:http://spss.com.

45. Novak, Miller, Houk. Kinematic Properties of rapid hand movements in a knob turning task. Exp Brain Res 2000;132:419-33.

46. Novak K, Miller L, Houk J. The use of overlapping submovements in the control of rapid hand movements. Exp Brain Res 2002;144:351-64.

47. Hurst A, Mankoff J, Dey AK, Hudson SE. Dirty Desktops: Using a Patina of Magnetic Mouse Dust To Make Common Interactor Targets Easier To Select. 20th Annual ACM Symposium on User interface Software and Technology UIST '07. Newport, Rhode Island, USA: ACM Press, New York, NY; 2007.

48. Morris, Grattan-Smith, Jankelowitz, Fung, Clouston, Hayes et al. Athetosis II: The Syndrome of Mild Athetoid Cerebral Palsy. Movement Disorders 2002;17(6):1281-7.

49. Sanger TD, Kaiser J, Placek B. Reaching movements in childhood dystonia contain signal-dependent noise. Journal of child neurology 2005;20(6):489-96.

50. Madhusudanan M. Dystonia : emerging concepts in pathophysiology. Neurol India 1999;47(4):263-7.

51. Gilio F, Curra A, Inghilleri M, Lorenzano C, Suppa A, Manfredi M et al. Abnormalities of motor cortex excitability preceding movement in patients with dystonia. Brain 2003;126(Pt 8):1745-54.

52. Ding D, Cooper RA, Spaeth DM. Isometric Joystick tuning interface and assessment. RESNA Annual Conference. Orlando, FL; 2004.