ABSTRACT

INTERACTION OF DISPARITY AND ACCOMMODATIVE VERGENCE

by
Michele Liu Kung

To fixate on a target that moves from far to near, changes in blur and disparity activate accommodation and disparity vergence. The goal of this study was to experimentally obtain eye movement data from four subjects, and analyze this data using through a new signal processing algorithm known as independent component analysis (ICA). Preliminary data suggest that three underlying neural subcomponents are present where the two components of disparity vergence initiate the movement and the accommodative portion is activated to facilitate the steady state portion of the response.

ICA was used as a blind source separation technique to analyze experimental and simulated data. Loss of independence between the sustaining component and the accommodative component is speculated to cause ICA to be unable to determine the accommodative component.

INTERACTION OF DISPARITY AND ACCOMODATIVE VERGENCE

by
Michele Liu Kung

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Department of Biomedical Engineering

August 2003

APPROVAL PAGE

INTERACTION OF DISPARITY AND ACCOMMODATIVE VERGENCE

Michele Liu Kung

Dr. Tara L. Alvarez, Thesis Advisor Assistant Professor Biomedical Engineering, NJIT	Date
Dr. Stanley S. Reisman, Committee Member Professor of Biomedical Engineering, NJIT	Date
Dr. Richard Greene, Committee Member Research Professor Biomedical Engineering, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Michele Liu Kung
Degree:	Master of Science in Biomedical Engineering
Date:	August 2003

Graduate and Undergraduate Education:

- Master of Science in Biomedical Engineering
  New Jersey Institute of Technology, Newark, New Jersey, 2003

- Bachelor of Science in Engineering Science
  New Jersey Institute of Technology, Newark, New Jersey, 2001

Major: Biomedical Engineering

Presentations and Publications:

M. Kung, T. L. Alvarez, J. L. Semmlow,
"Interaction of Disparity and Accommodative Vergence",
Proceedings of the IEEE 29th Annual NorthEast Bioengineering Conference,
Newark, NJ, Mar. 2003.

A. Daftari, T. L. Alvarez, M. Kung, J. L. Semmlow,
"Neuro-Control in Divergence Eye Movements",
Proceedings of the IEEE 29th Annual NorthEast Bioengineering Conference,
Newark, NJ, Mar. 2003.

A.C. Ranzini, C.V. Ananth, J.C. Smulian, M. Kung, A. Limbachia, A.M. Vintzileos,
"Ultrasonography of the fetal thyroid: nomograms based on biparietal diameter and gestational age" J Ultrasound Med. 2001 Jun; 20(6):613-7.

To Ven and Lily Kung

ACKNOWLEDGMENT

I would like to express my sincerest thanks to Dr. Tara L. Alvarez, my advisor, for providing valuable and countless resources, insight, and intuition as well as support, encouragement and reassurance. Special thanks to Dr. Stanley Reisman for advisement, assistance and for participating actively in my committee. Special thanks to Dr. Richard Greene for participating in my committee. I would like to acknowledge the time and encouragement that I received from Dr. John Semmlow.

My fellow students in the Vision and Neural Engineering Laboratory at NJIT are deserving of recognition for their support. Thanks also for providing such an excellent working environment. I would like to express my gratitude for my dedicated subjects for their time and efforts. Special thanks to Florence Chua, for being my first and most patient subject during data collection.

My deepest gratitude to my parents, Ven W. Kung and Lily Y. Kung, for their love, patience, and support. They have always kept my success as their dream.

I am very grateful to Berzelius Y. Cada for his support, encouragement and especially his proofreading.

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(continued)

LIST OF FIGURES
(continued)

Figure **Page**

LIST OF FIGURES
(continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Vergence Eye Movements

There are five different types of eye movements, saccades, pursuit, vergence, vestibular and fixation maintenance [1]. Only one of these types of eye movements, vergence, is disjunctive or disconjugate which means the eyes do not move in tandem with each other. During a vergence or disconjugate eye movement, the eyes simultaneously move in opposite directions. A convergence eye movement occurs when the eyes move laterally toward the nose (cross-eyed). A divergence eye movement occurs when the eyes move laterally away from the nose. A typical situation calling for a disjunctive movement is the change in binocular observation from a far to a near target, when the fixation lines of the two eyes have to converge. A target changing in depth produces primarily blur and disparity that evoke both accommodative and vergence eye movements respectively. Accommodative vergence is driven by blur. The accommodation system drives the lens to focus an image. The gradual hardening of the lens, a natural process of aging, causes the accommodation system to cease functioning over time. This natural deterioration usually starts around the age of thirty-five and is known as presbyopia [1]. Disparity vergence occurs when images fall on opposite sides of the fovea (part of the retina), causing diplopia (double vision) [1]. The eyes rotate inward or outward to place the image on the fovea, which fuses the two images into one. This retinal disparity is one of the key inputs into the disparity vergence control system. By controlling the conditions of the stimulus we are able to stimulate disparity vergence eye movements with constant accommodation, as well as disparity vergence eye movements with a change in accommodation [2].

It is also possible to eliminate blur and drive the vergence motor response with disparity alone by controlling the stimulus conditions[3].

1.1.1 Physiology

The eye moves through the use of six extraocular muscles that attach to each eyeball and perform horizontal movements, vertical movements or rotation. These muscles are controlled by impulses from specific cranial nerves that tell the muscles to contract or to relax. When certain muscles contract, and others relax, the eye moves. The six muscles that control the movement of the eyes are the superior rectus, the inferior rectus, the superior oblique, the inferior oblique, the lateral rectus, and the medial rectus shown in Figure 1.1. The superior rectus primarily moves the eye upward and secondarily rotates the top of the eye toward the nose. The inferior rectus primarily moves the eye downward and secondarily rotates the top of the eye away from the nose. The superior oblique primarily rotates the top of the eye toward the nose and secondarily moves the eye downward. The inferior oblique primarily rotates the top of the eye away from the nose and secondarily moves the eye upward. The medial rectus moves the eye toward the nose and the lateral rectus moves the eye away from the nose [1].

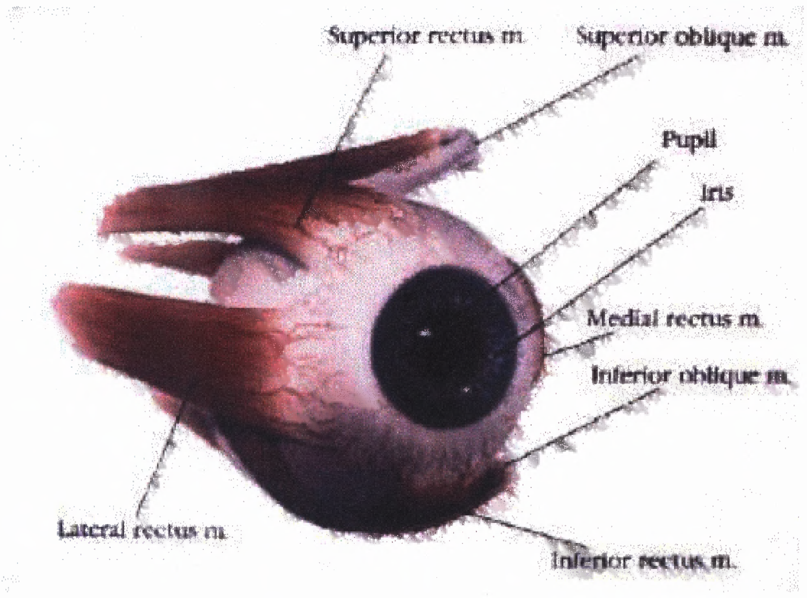

Figure 1.1 Muscle anatomy of the eye [4].

1.1.2 Dual Mode Theory

The disparity vergence eye movement system has been shown to exhibit complex dual mode slow and fast components within responses. The neurophysiology of these disjunctive movements has been shown to be neurologically independent from conjugate eye movement [5] (saccadic, pursuit, vestibular, and fixation maintenance). Mays et al were able to describe the activities of neurons that burst for disjunctive eye movements. The activity of these burst cells were correlated with vergence velocity, and size of the vergence movement [5]. The time characteristics allow correction of movements during their course on the basis of visual information [6]. Semmlow et al isolated signals of pure disparity vergence responses that match these physiological findings [7]. The Dual Mode Model has been developed which simulates this behavior [8]. Disparity vergence uses dual control to obtain both speed and accuracy. The Dual Mode Theory is a two-component system, comprised of an initiating and a sustaining component. The initiating component is a feed forward, open loop control system, which depicts the system's speed. The sustaining component is a feedback, closed-loop control system, which depicts the system's accuracy. The initial transient movement of vergence often differs by as much as a degree or more from the one required for binocular fixation. This error is subsequently corrected by slow changes in convergence [3]. These slow changes in convergence have been correlated to vergence tonic cells [9].

This model has been validated by neurophysiological data, which shows burst and tonic cells exist in the vergence neural circuit. The burst cells correlate to the feed forward or pulse portion of the vergence model and the tonic cells correlate to the feedback or step portion of the vergence model [5].

Hung and colleagues studied the difference between disparity vergence with constant accommodation and disparity vergence with a change in accommodation. Hung and colleagues studied variance and showed that a significant amount of variance existed during the transient phase after response latency, which corresponds to the two components found in disparity vergence. It is hypothesized that there is a third feedback driven component from the accommodative vergence system [10]. However, their key finding was in comparing disparity vergence with constant accommodation versus disparity vergence with a change in accommodation. More variance was found in the later part of the disparity vergence with a change in accommodation compared to disparity vergence with constant accommodation suggesting that accommodation is present during the later steady state portion of the response [11].

1.2 Independent Component Analysis

Independent component analysis (ICA) is an advanced statistical technique for decomposing complex data set vector into components that maximize statistical independence between components [12, 13]. ICA can be used to solve blind source separation (BSS) problems. ICA assumes that the components are statistically independent, have non-gaussian distributions and are linearly mixed [7]. In mathematical notation this problem can be modeled by the equation $X = As$ + noise. Where X is the mixed signal matrix where there are at least as many signals as there are sources, A is the mixing matrix, and s is the component matrix.

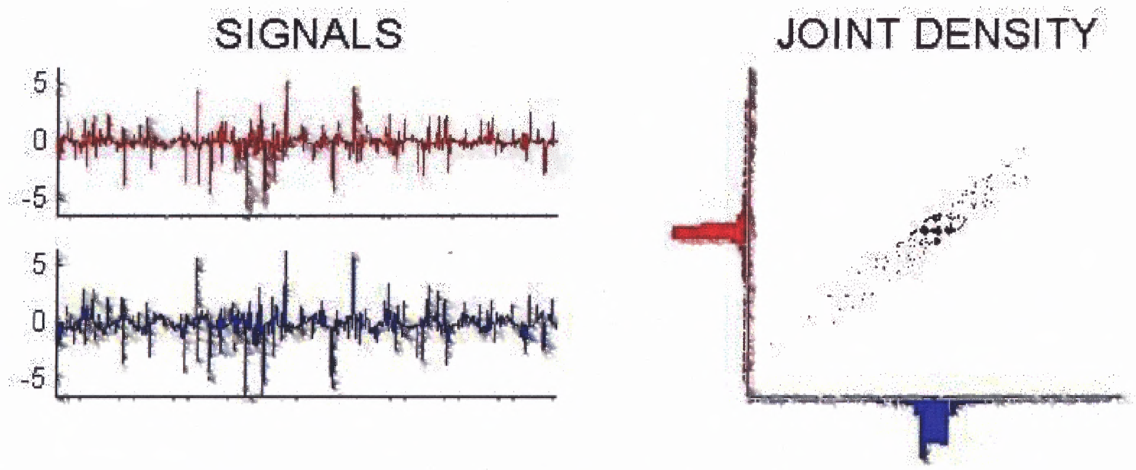

Figure 1.2 Sample input signals and signal density [14].

Most ICA algorithms go through a preliminary whitening or sphering of the data X. Sphering is accomplished by linearly transforming the original observed variable (v) to X ($Qv = X$), such that $E\{XX^T\} = I$. After this whitening process, the separate signals can be found by orthogonally transforming the whitened signals. This is achieved simply by rotation of the joint density and scaling the data set. The appropriate rotation is sought by maximizing the nongaussian-ity of the marginal densities (shown on the edges of the density plot) [13, 14]. The linear mixture of independent random variables is necessarily more Gaussian than the original variables.

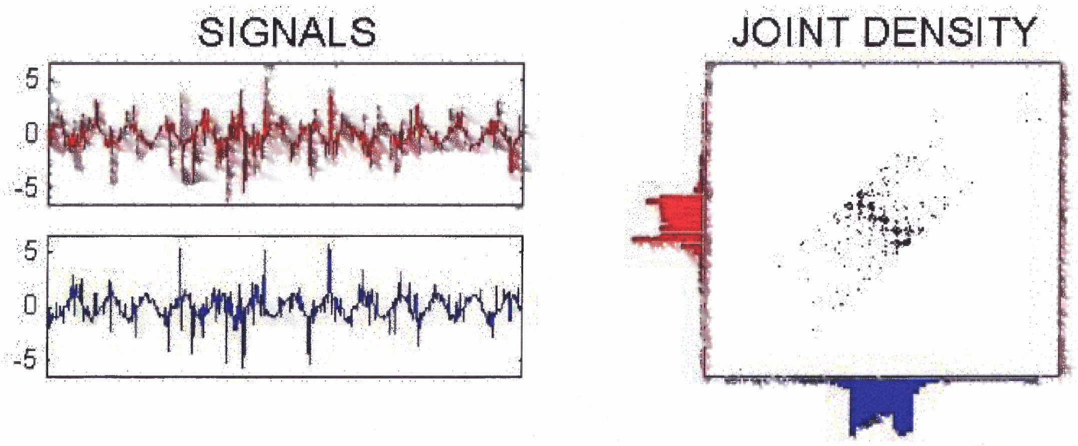

Figure 1.3 Signal and joint density after whitening (sphering) operation [14].

The ICA technique requires a number of repetitive responses for the behavior being analyzed. When ICA is applied to ensemble vergence movement data, it treats each response (observation) run as a separate mixed signal [7]. There are many popular ICA algorithms available online as MATLAB script files. This thesis utilized the "FastICA" algorithm developed by the ICA Group at the Helsinki University.

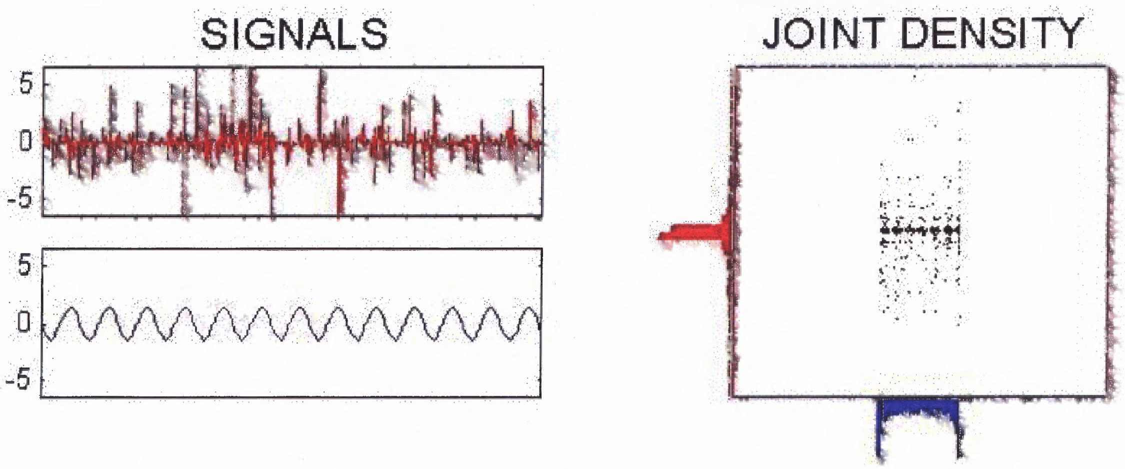

Figure 1.4 Separated signals after five steps of FastICA [14].

1.2.1 Independent Component Analysis : Application

An interesting application of ICA used as a blind source separation (BSS) technique is the classic cocktail party problem. In this application multiple people are speaking simultaneously in the same room. The problem is to separate the different voices using recordings made by multiple microphones placed throughout the room. As long as there are at least as many microphone recordings as components ICA can be applied to separate the individual voice signals. By using an ICA algorithm researchers are able to separate the various voices into separate signals. Another more practical example of this application of ICA would be noise reduction [13].

Principal component analysis (PCA) is a technique used for estimating the true dimensionality of a data set. Principal component analysis works by generating components that account for as much variability in the data set as possible. PCA also generates a series of eigen values to describe how much of the data's variance is contained in that component. A Scree plot (eigenvalue against component number) is often used to determine how many components ICA should determine. Semmlow et al determined that for disparity vergence two components represent the majority of variability of the data based on the knee or breakpoint in the scree plot [2]. The FastICA package can also be used to generate a plot of eigenvalues versus number of component to estimate how many components the ICA should blindly separate [15].

1.3 Objective

The objective of this study was to experimentally obtain eye movement data from four subjects, and analyze this data using independent component analysis. Two types of controlled stimulus were used to collect this data. The first stimulus type evoked disparity vergence with constant accommodation. The second stimulus type evoked disparity vergence with a change in accommodation. Independent component analysis was used to find the hypothesized accommodative component [11]. This thesis advanced the understanding of the interactions between the disparity vergence system and the accommodative vergence system. This knowledge could potentially help people with accommodative and disparity dysfunction (ex presbyopia).

CHAPTER 2

SIMULATIONS

2.1 Simulated Data Creation

Independent component analysis has already been successfully applied to disparity vergence data with constant accommodation in multiple studies [2, 7]. Semmlow et al were able to verify the use of ICA on vergence movement data by comparing ICA results on simulated data with ICA results on experimental data (disparity vergence data with constant accommodation). Simulated data was created based on the Dual-Component model by randomly varying the component width, amplitude and onset time.

For this thesis, two sets of simulated disparity vergence data were created using a MATLAB script [16]. The program created twenty simulated vergence movements using the open loop parameter with zero noise. The inputs shown in table 2.1 were used to create simulated data with the accommodative component suppressed to zero shown in figure 2.1A. The simulated data created using these inputs were created to be equivalent to four degree experimental haploscope data taken for two seconds at two hundred Hz. This simulated data (shown in figure 2.1B) was used to reproduce Semmlow et al's work with ICA to two components [7].

Table 2.1 Inputs given to MATLAB script used to create simulated data with suppressed accommodative component.

Parameter: Default Value	Value Used	Max Variation
(1) IC Lat: 0.17	0.2	.1
(2) Slew Rate : 10	5	2
(3) Step Gain : 1	1	0.1
(4) Pulse Gain: 0.22	0.22	0.1
(5) FBK Gain: 1	1	0
(6) SC FBK gain: 0	0	0
(7) Step Lat : 0.27	0.2	.1
(8) SC Gain : 0.1	0	0
(9) SC Lat : 0.4	0.4	0
(10) SC Derv Tau : [-12]	-12	0
(11) SC Slew : 30	30	0
(12) Plant Tau : [1 4]	[1 4]	0

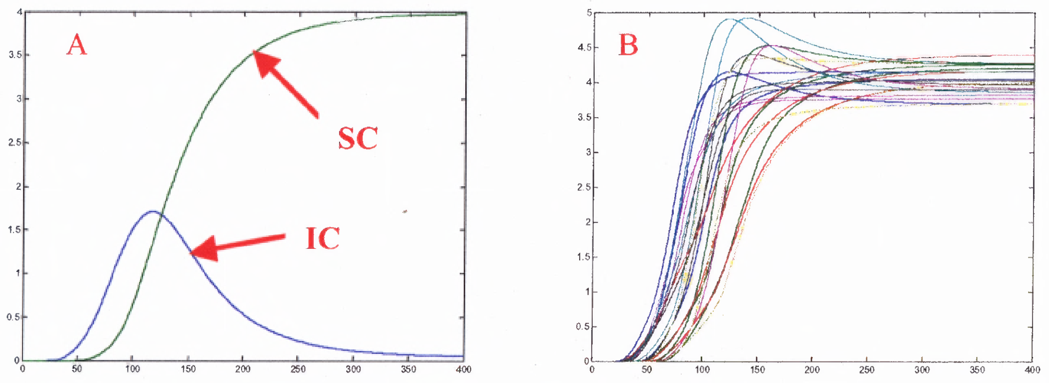

Figure 2.1 (A) Simulated data: plot of individual means of the sustaining component (SC) and initiating component (IC) used to create the simulated data shown in figure 2.1B [16]. (B) Simulated data: four degree vergence data created in MATLAB. Two seconds at 200 Hz. (degrees vs. sample number) [16].

The inputs shown in table 2.2 were used to create simulated data with the accommodative component included. The simulated data created using these inputs were created to be equivalent to four degree experimental LED data taken for two seconds at two hundred Hz with zero noise. This data includes the hypothesized third feedback driven component from the accommodative vergence system [10].

Table 2.2 Inputs given to MATLAB script used to create simulated data with variable accommodative component.

Parameter: Default Value	Value Used	Max Variation
(1) IC Lat: 0.17	0.17	0
(2) Slew Rate : 10	10	0
(3) Step Gain : 1	1	0.1
(4) Pulse Gain: 0.22	0.22	0.1
(5) FBK Gain: 1	1	0
(6) SC FBK gain: 0	0	0
(7) Step Lat : 0.27	0.2	0
(8) SC Gain : 0.1	0.3	0
(9) SC Lat : 0.4	0.4	0
(10) SC Derv Tau : [-12]	-12	0
(11) SC Slew : 30	30	0
(12) Plant Tau : [1 4]	[1 4]	0

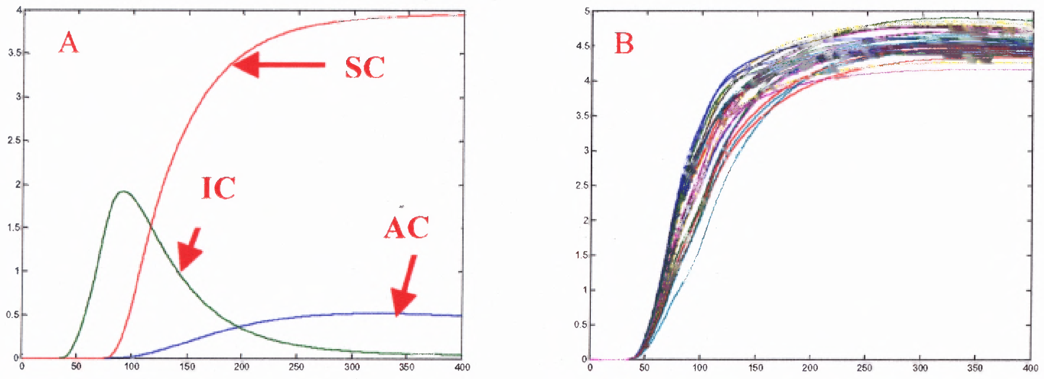

Figure 2.2 (A) Simulated data: plot of individual means of the sustaining component (SC), initiating component (IC) and hypothesized accommodative component (AC) used to create the simulated data shown in figure 2.2B [16]. (B) Simulated data: four degree LED created in MATLAB. Two seconds at 200 Hz. (degrees vs. sample number) [16].

2.2 Data Analysis

Independent component analysis was run on the simulated data using the icamtl.m file written by Semmlow et al, and customized for the purpose of this thesis (see Appendix B for excerpt). The icamtl algorithm used the FastICA algorithm, but was different from using FastICA alone in that the icamtl algorithm could only be run on data of two seconds (400 points) and data could be partitioned to help in the ICA to remove the issues of dependence between components that start at approximately the same time [17]. Another difference is the limitation of the icamtl algorithm to only determining two components. The correction algorithm is not required for real (experimental) data, sufficient independence exists between components of real data. This greater independence is likely the result of a greater number of variables controlling the physiological system than the model simulation [2].

The FastICA algorithm was also used to blindly separate two components on the simulated LED data to compare to ICA performed to two components on experimental LED data. FastICA to three components was performed on the simulated LED data to compare to ICA performed to three components on experimental LED data.

2.3 Results

2.3.1 Icamtl on Simulated Data

Figure 2.3 shows the results of the successful reproduction of ICA work performed by Semmlow and his colleagues on simulated data [7]. Figure 2.3A shows the ensemble averages of the simulated motor components. Figure 2.3B is a plot of the components determined by ICA. Figure 2.3C shows both sets of components (means of simulated component input and ICA determined components) plotted on the same graph. This analysis was considered successful due to the similarities seen in the components plotted in figure 2.3C. This successful reproduction of analysis performed by Semmlow et al confirmed that ICA was working properly [7].

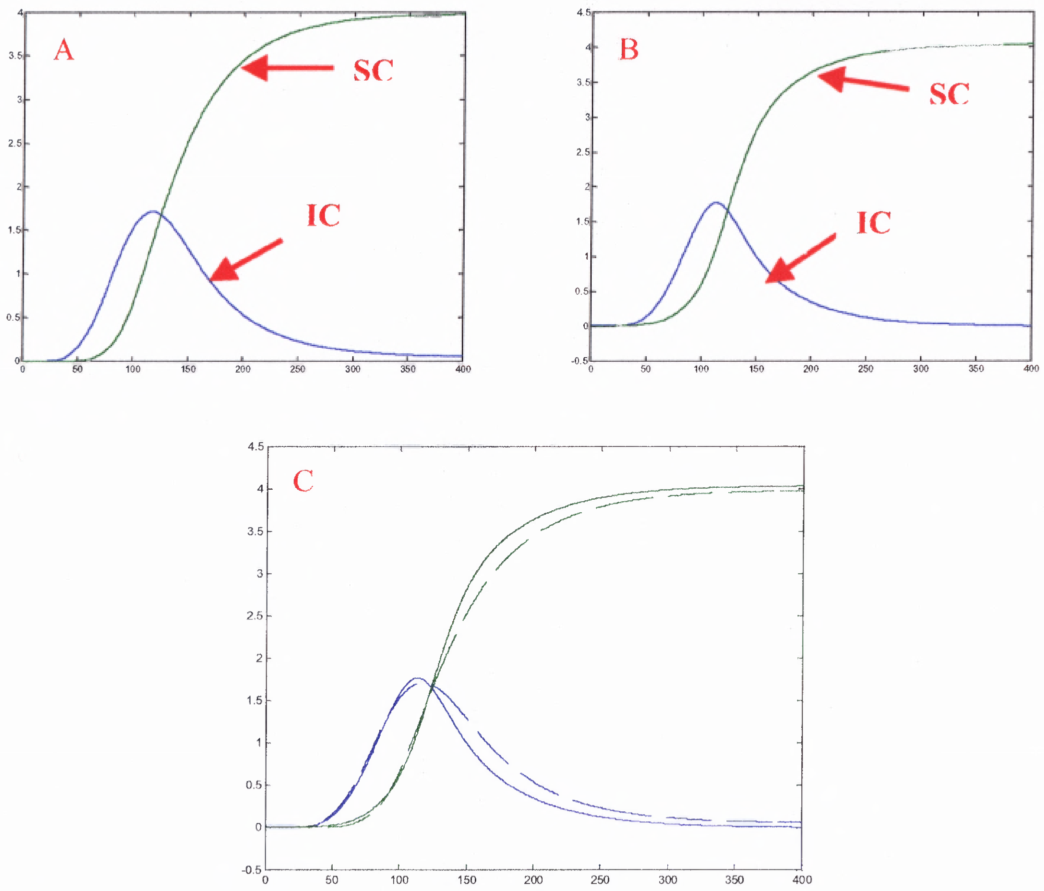

Figure 2.3 Simulated data: (A) Plot of individual means of the sustaining component (SC) and initiating component (IC) used to create the simulated data shown in figure 2.1B [16]. (B) Components determined by icamt1.m algorithm written by Semmlow et. al. Partitioned at 119. (C) Simulated components (solid lines) used to create inputted simulated data plotted with components determined by ICA (dashed lines).

Figure 2.4A shows the means of the components used to create the simulated LED data in figure 2.2B. Figure 2.4B displays the two components determined by the icamt1 algorithm written by Dr. Semmlow and his colleagues [17]. Figure 2.4C has the simulated components and the ICA determined components plotted on the same graph to show the differences between the input components and the components determined by ICA.

These differences result from the limitation of the icamt1 algorithm to two component determination. The sustaining component that icamt1 determined is a combination of the sustaining component and the accommodative component input into the algorithm. The icamt1 algorithm scales the components by holding the value of the steady state portion of the sustaining component (SC) found to the value of the average of the data input into the algorithm seen in figure 2.4A. The algorithm then scales the initiating component (IC) found such that the two components determined by ICA add up to the average of the input data. Because of the scaling, the SC found has a larger magnitude than the input SC and consequently the IC found is smaller than the input IC.

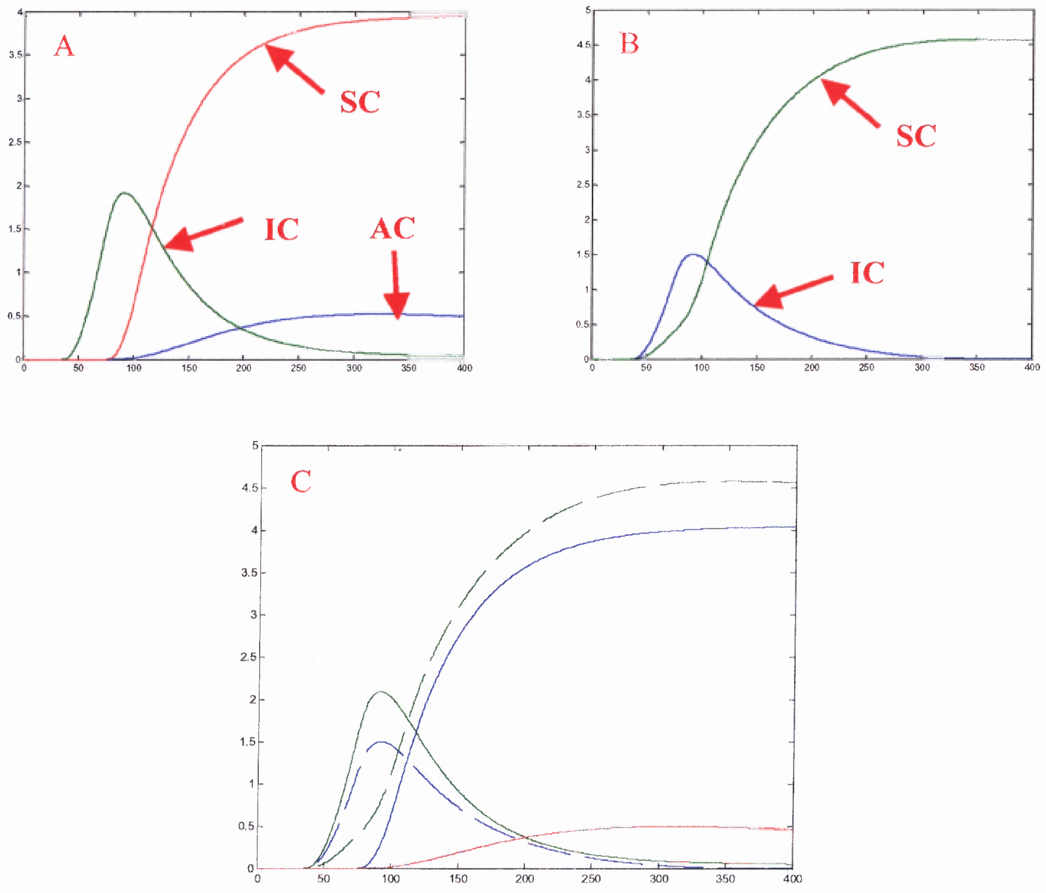

Figure 2.4 Simulated data: (A) Plot of individual means of the sustaining component (SC), initiating component (IC) and accommodative component (AC) used to create simulated data shown in figure 2.2B. (B) Components determined by icamt1.m algorithm written by Semmlow et. al. Sustained component (S.C) and initiating component (I.C) labeled. Partitioned at 110 [17]. (C) Simulated components (solid lines) used to create inputted simulated data plotted with components determined by ICA (dashed lines).

2.3.2 FastICA on Simulated LED Data

The sloping down of both components in figure 2.5B is speculated to be caused by the third component that ICA was not told to determine for this analysis. The three components that the independent component analysis (ICA) found (shown in figure 2.5C) did not match the three components that were input into the simulated data (shown in figure 2.5A). The third component found of the simulated data was multiplied by negative one if to scale the end of the third component positive. The most likely explanation for why the ICA data on this simulated LED data did not return the components entered is due to loss of independence of the three components. This loss of independence is probably caused by the similarities in shape between the accommodative component (AC) and the sustaining component (SC). Both the AC and the SC are steps although the accommodative component has a longer latency and is of a smaller magnitude.

ICA algorithms do not find the proper scale of the components, it defines the behavior of the components. Principal component analysis (PCA) determines the number of components. ICA will search for the number of components the operator tells the algorithm to search for. While ICA was able to find a step component that looks like the SC and a pulse component, it can be observed that the remaining component found (figure 2.5C) is a mixture of the actual input components (figure 2.5A). Thus it is hypothesized that the third component that ICA determines is a combination of the primary and secondary components. This component has an initial pulse shape, and a latter upwards drift giving it an 'N' shape.

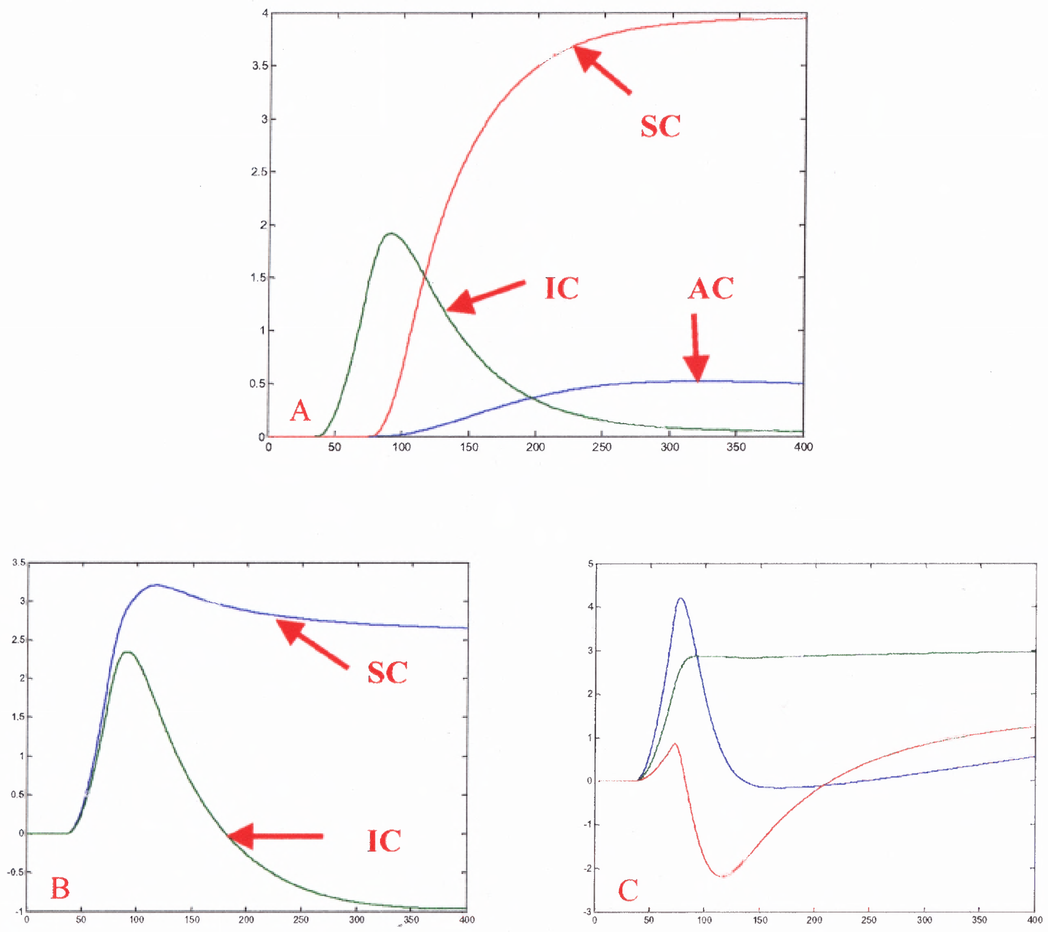

Figure 2.5 Simulated four degree LED data: (A) Plot of individual means of the sustaining component (SC), initiating component (IC) and accommodative component (AC) used to create simulated four degree LED data. (B) ICA on simulated four degree LED data to two components. Initiating component (I.C) and sustaining component (S.C) labeled (degrees vs. sample number). (C) ICA on simulated four degree LED data to three components (degrees vs. sample number).

CHAPTER 3

EXPERIMENTAL DATA

3.1 Experimental Methodology

3.1.1 Experimental Instrumentation

The Skalar Iris model 6500, an infrared limbus-tracking device, recorded data at a sampling rate of 200 Hz for both experiments. This eye movement monitor has a resolution of 2 minutes of arc and a linearity of $\pm$25 degrees [18]. This instrument was placed on the subject's head and adjusted to the left and right eye. It collected data from each eye where left and right eye movements were individually stored to be analyzed offline. A lens kit mounted onto the eye movement monitor was used to compensate for myopia (nearsightedness) when needed.

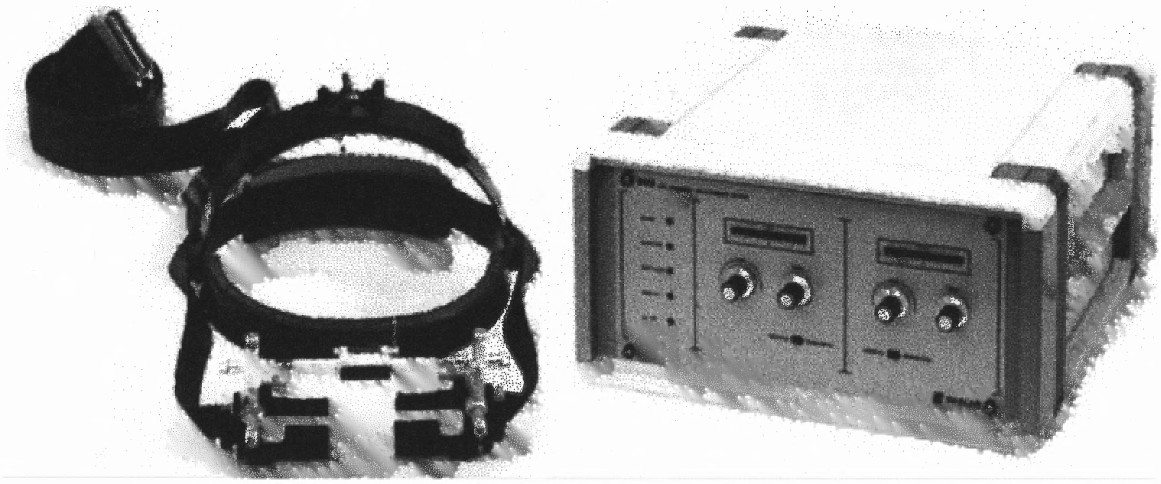

Figure 3.1 The Skalar Iris Limbus Tracker model 6500 [18].

A Dell Optiplex GX240 Pentium IV, 1.70 GigaHertz (GHz) computer with 256 MB of RAM operating with Microsoft Windows XP Professional 2002 ran LabVIEW 6i. This computer contained a data acquisition (DAQ) board, model 6024e, from National

Instruments, which digitized incoming analog signals as voltage values and output analog signals to the stimulus display which were a pair of analog oscilloscopes (BK Precision Model 2120B 30MHz) and digital outputs to illuminate several light emitting diodes (LEDs). Throughout the experiment, the data were collected from each eye independently. The net vergence response is taken as the difference in the position of the two eyes and is computed from the individual eye movement recordings [7] where convergence is plotted as positive. For these experiments, a four/six degree vergence movement consisted of each eye moving laterally two/three degrees from the initial position to the final position of each trial. Therefore the net vergence response equals the total degrees moved (four or six).

After calibration, subjects would push a trigger button to initiate an experiment with a random time delay of 0.5 to 2.0 seconds (either LEDs or oscilloscopes) to remove anticipation by the subject. Furthermore, the four and six degree steps were randomly selected by the computer to avoid subject prediction. These responses were recorded for three seconds and experiments occurred in complete darkness where the subject only observed the presence of the targets to eliminate proximal vergence cues.

3.1.2 Experimental Setup – Oscilloscopes

The apparatus for the oscilloscope set up (haploscope) is shown in figure 3.2. It consists of two partially reflective mirrors positioned 45 degrees to the subject's line of sight, two oscilloscopes that provide the step stimulus, and a limbic tracking system, which collects the eye movements.

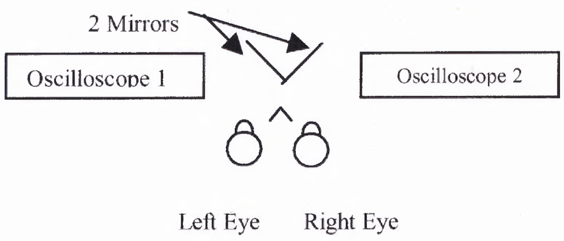

Figure 3.2 Haploscope setup.

Each oscilloscope emits a line stimulus towards the mirrors, which in effect produce two lines that the subject would fuse into single line. After the subject pushed the trigger, the two lines would move in a step manner triggering either a four degree or six degree convergence response (an inward turning of the eyes where the person would "cross their eyes"). The subject would follow the stimulus and would once again fuse the lines. The oscilloscopes project targets at a constant focal length from the subject to stimulate disparity vergence. Because the targets are projected at a constant focal length from the subject, this experimental set up keeps accommodation constant due to lack of change in blur.

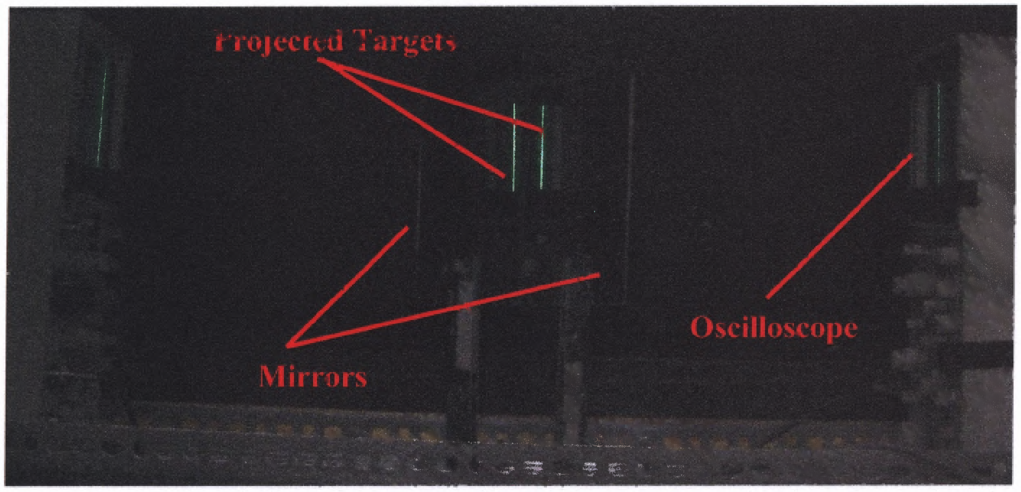

Figure 3.3 Subject view of haploscope setup.

3.1.3 Experimental Setup – LED

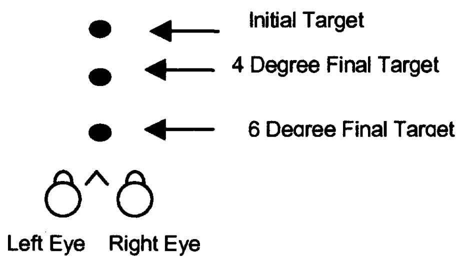

Figure 3.4 LED target setup.

For the three-dimensional (3-D) target LED setup shown in Figure 3.4, the stimulus always began at the same initial position using an illuminated LED. This setup consists of the target LEDs and the limbus tracking device. The computer initiated the extinguishing of the initial LED and the lighting of the next LED where subjects were asked to track the new target positioned along the midline of the subject as shown in Figure 3.4. Because the targets change in three dimensional space, the focal length of the target to the subject changes. This change in focal length stimulates disparity vergence with a change in accommodation.

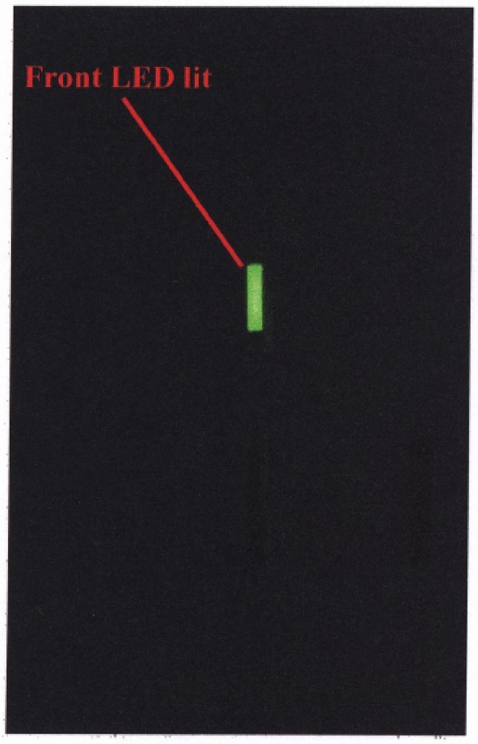

Figure 3.5 Subject view of LED setup. Front LED lit during calibration.

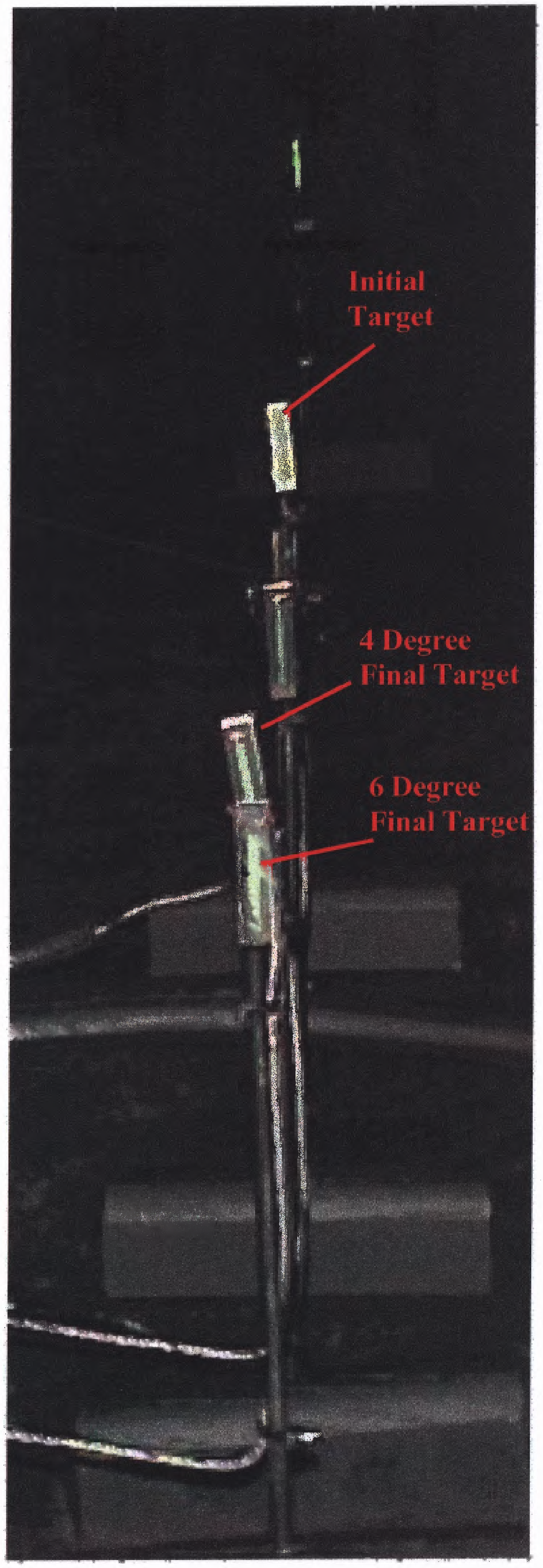

Figure 3.6 Top angled view of LED setup. Back LED lit during calibration.

3.1.4 Subject Selection

Four subjects were chosen to participate in this study, B01, M02, F03 and D04. Each subject signed informed consent form *"NJIT Consent Form_CI_ without funding group1 to 3"* (see appendix A) approved by the NJIT Institution Review Board. Subjects were all under the age of thirty and had normal binocular vision. People 35 years and older typically have presbyopia which is a condition where their lens is not as functional and thus were not included in this study.

3.1.5 Data Acquisition

Data acquisition was performed using a Labview Virtual Instrument (VI) written by other members of the vision research lab [19]. This program controlled the output to the experimental setups, as well as randomly selecting four or six degree stimuli.

3.2 Experimental Data

For comparison, experimental data are organized by experiment type and stimulus. Each vergence signal plotted is a separate experimental trial. The experimental data are plotted in degrees as a function of sample number (time). Experimental data were taken for each trial for three seconds at two hundred hertz (Hz) which is above the Nyquist sampling rate for vergence eye movements.

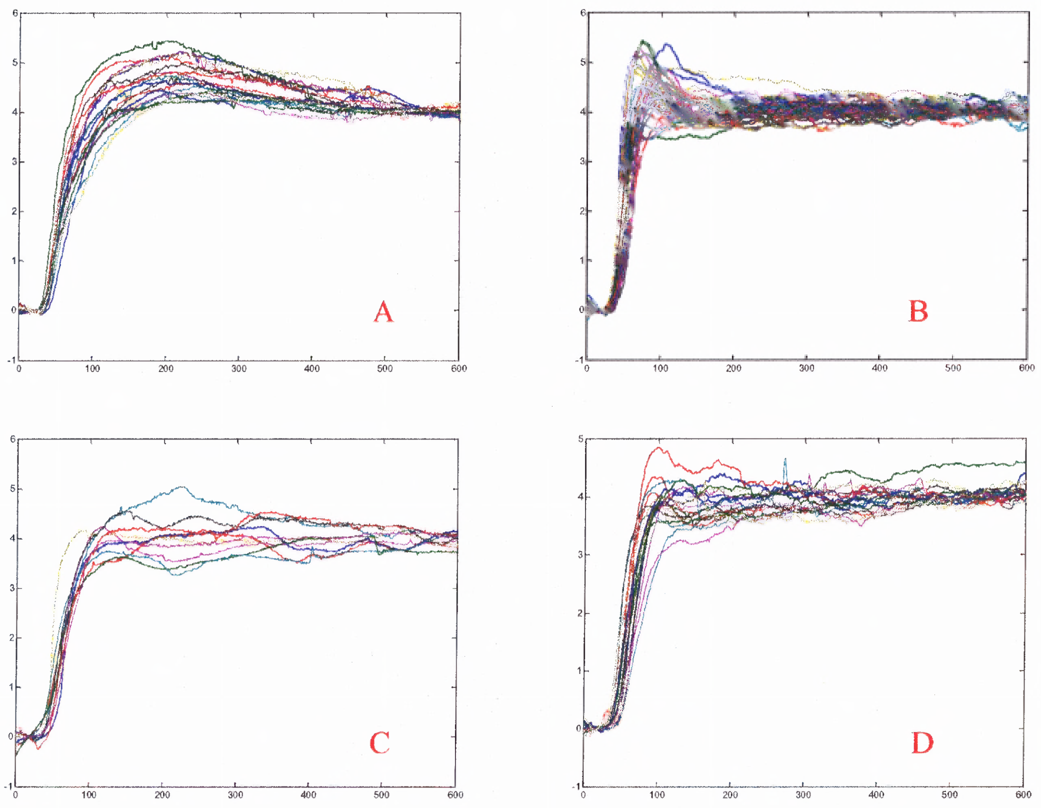

Figure 3.7 Four degree haploscope data. Three seconds at 200 Hz (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

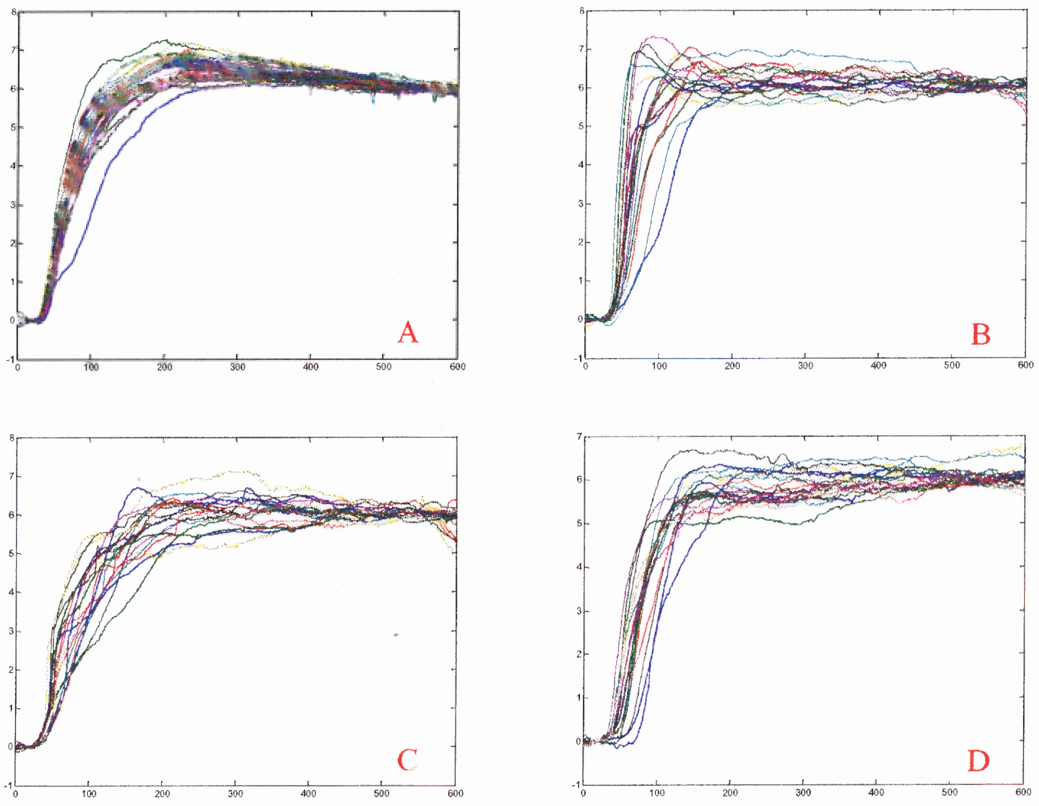

Figure 3.8 Six degree haploscope data. Three seconds at 200 Hz (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F02 (D) Subject D04.

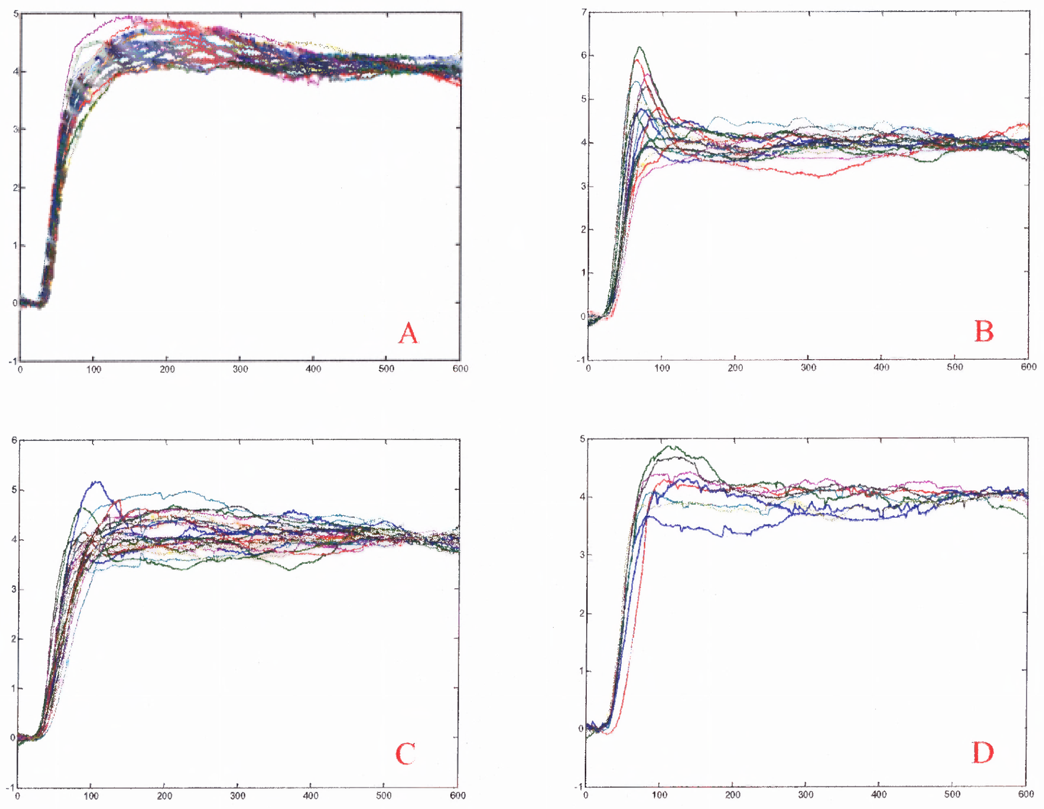

Figure 3.9 Four degree LED data. Three seconds at 200 Hz (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

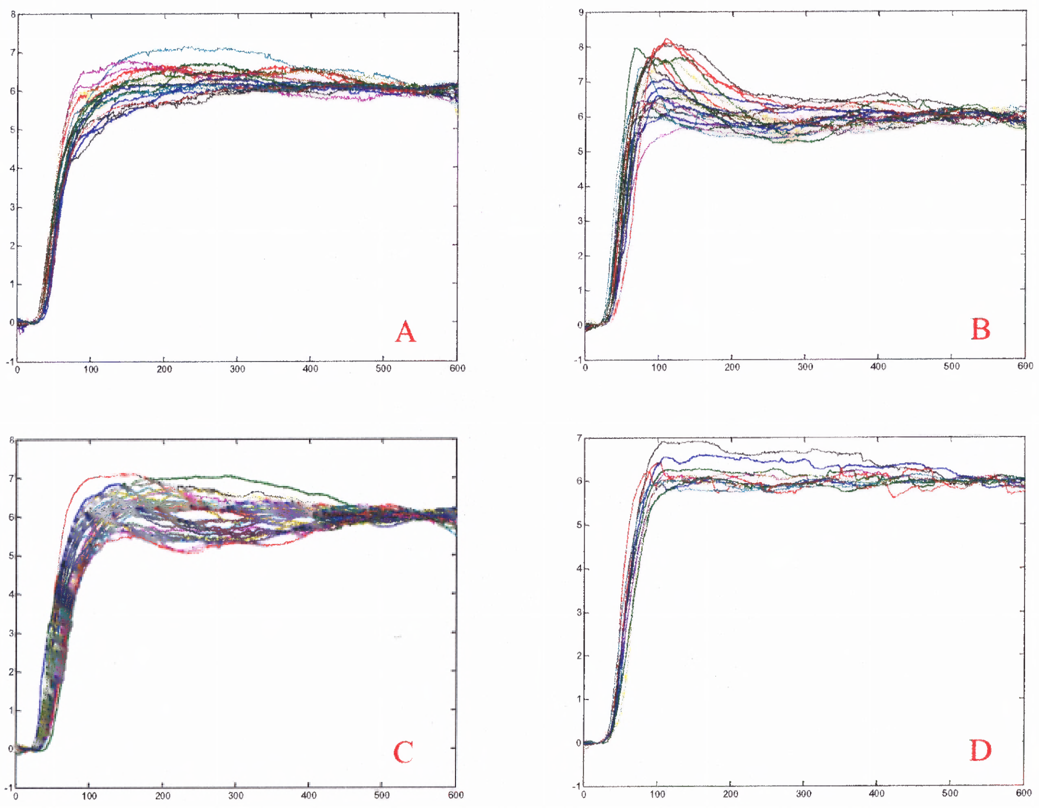

Figure 3.10 Six degree LED data. Three seconds at 200 Hz (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

3.3 Analysis

Principal component analysis (PCA) has shown that two components account for more than 90% of the variability in convergence eye movements. It has been hypothesized that the accommodative component (AC) of vergence and noise will account for a portion of the remaining 10% of the variability [11]. Semmlow et al has used independent component analysis (ICA) to find the sustaining component (SC) and the initiating component (IC) of disparity vergence data with constant accommodation [7].

This thesis used a custom vision data analysis program written in MATLAB [20] to categorize the vergence eye movement data. Only data with pure vergence movements were kept for analysis. Vergence movements with saccades, blinks or excess instrumentation noise were discarded. This program was also edited to save the vergence eye movement data to be used in the independent component analysis (see visresc2call.m in appendix B). The data analysis program was run under MATLAB 6.1. After the data were categorized, they were further sorted using MATLAB 6.5 scripts created for this thesis (see combine.m, sorta.m, sortb.m in appendix B).

ICA was performed on the categorized vergence data. The vergence movement data were truncated from three seconds (600 points) to two seconds (400 points) using a MATLAB script written for this thesis (see truncate.m in appendix B). The icamt1.m file written by Semmlow et al, and customized for the purpose of this thesis (see Appendix B for excerpt) was run on the truncated experimental data. The icamtl algorithm used the FastICA algorithm, but were different from using FastICA alone because ICA could only be run on data of two seconds (400 points) and data could be partitioned to help in the ICA to remove the issues of dependence between components that start at approximately the same time [17]. This correction algorithm is not required for real (experimental) data, sufficient independence exists between components of real data. This greater independence is likely the result of a greater number of variables controlling the physiological system than the model simulation [2].

The FastICA package [15] written for MATLAB was run on the full (not truncated) vergence movement data. The FastICA package was selected because it has been found to converge reliably when performed on vergence data [7]. The FastICA script was run with other MATLAB scripts created for this thesis (see f_loaderm.m, f_saverm.m, icaf_loaderm.m in appendix B). This analysis was run under MATLAB 6.5. ICA was run to determine the two components on the haploscope data and to determine the three components on the LED data.

The components found by FastICA are not to scale. The third component found in the LED data was multiplied by negative one if needed to match the direction of the third component found in the simulated data. ICA performed on LED data to three components was compared with ICA performed to three components on the simulated data. The ICA performed to three components on the simulated data did not match the three components that were used in creating the simulated data.

3.3.1 Icamt1 – Four Degree Movement

In most of these graphs a downward curve is seen as the primary difference between the haploscope and the led data. It has been speculated that this deviation in the step component is a result of the accommodative vergence response [21].

In figure 3.11a the sustaining component is completely flat, which supports the speculation that the downward curve of the sustaining component seen in figure 3.11b is caused by the influence of accommodative vergence response [21].

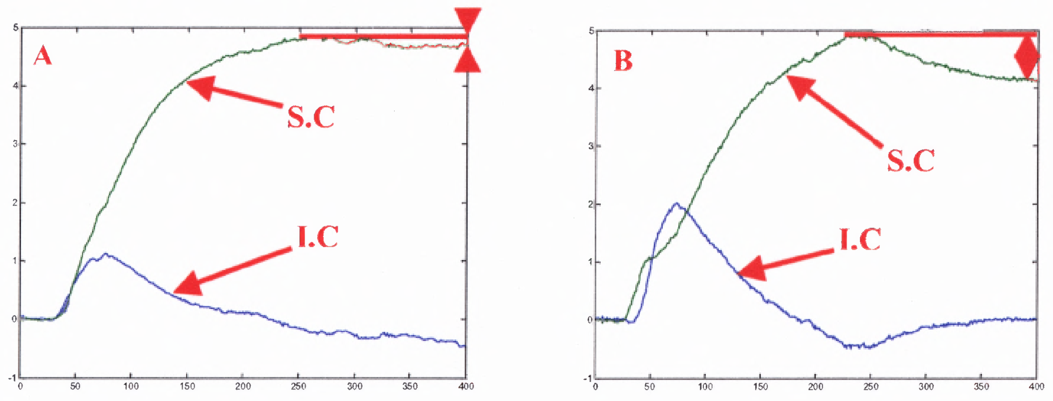

Figure 3.11 Subject B01: (A) Four degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 70 [17]. (B) Four degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 75 [17].

In figure 3.12 there is not really a visible downward curve difference between the Haploscope data, and the LED data. It is hypothesized that for some people the accommodative response is slower than for others, and in this case the accommodative response has not started yet.

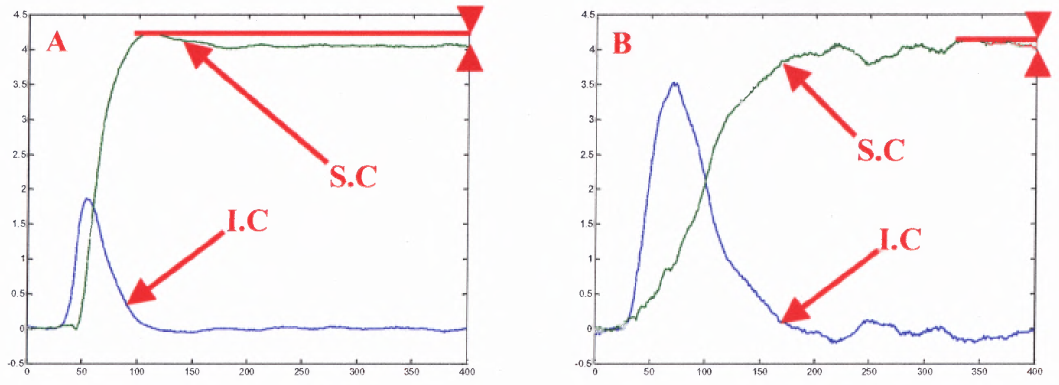

Figure 3.12 Subject M02: (A) Four degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 55 [17]. (B) Four degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 85 [17].

In figures 3.13a and 3.14a there is visibly less downward curve than in figures 3.13b and 3.14b. Again this effect is speculated to be caused by the accommodative vergence component.

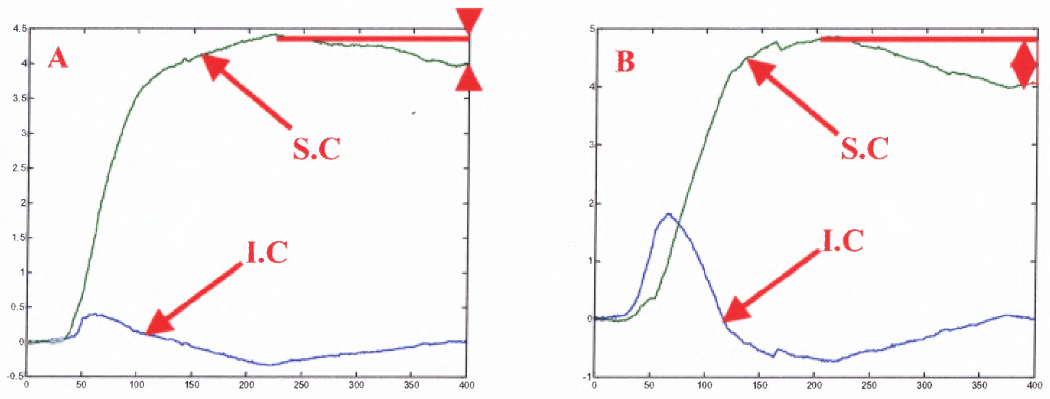

Figure 3.13 Subject F03: (A) Four degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 65 [17]. (B) Four degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 75 [17].

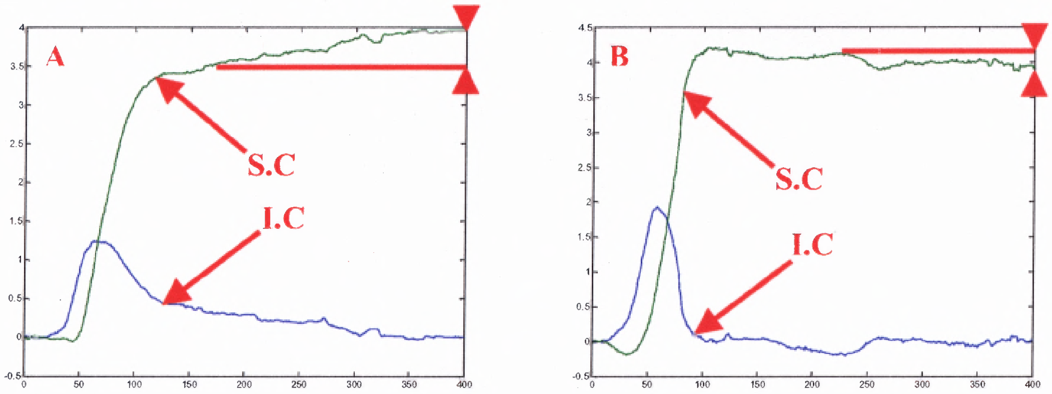

Figure 3.14 Subject D04: (A) Four degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 70 [17]. (B) Four degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 65 [17].

3.3.2 Icamt1 – Six Degree Movement

In this section the icamt1 analysis on the six degree vergence data is shown. The sustaining component for the haploscope data in this section remained flat in the steady state portion (figures 3.15A, 3.16A, 3.17A, 3.18A). There is less of a downcurve in the sustaining component in the steady state portion of the LED component graphs shown in this section (figures 3.15B, 3.16B, 3.17B, 3.18B) compared to the four degree analysis shown in section 3.3.1 (figures 3.11B, 3.12B, 3.13B, 3.14B). This might be caused by a higher latency for the accommodative response for this larger stimulus.

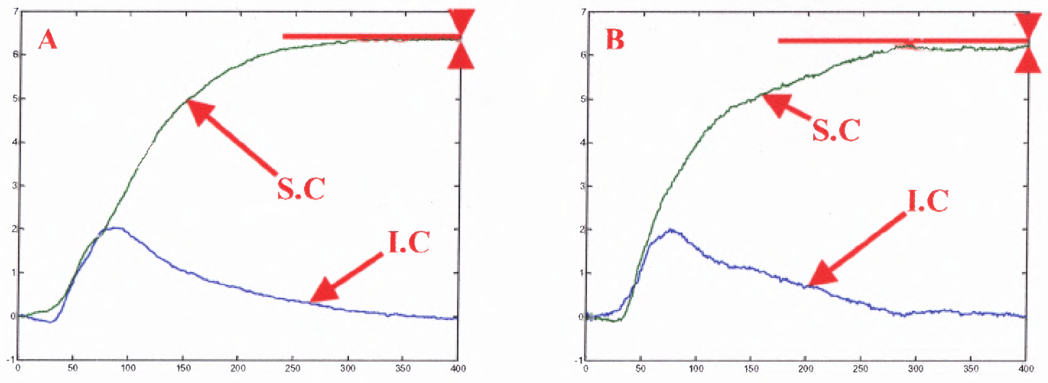

Figure 3.15 Subject B01: (A) Six degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 95 [17]. (B) Six degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 60 [17].

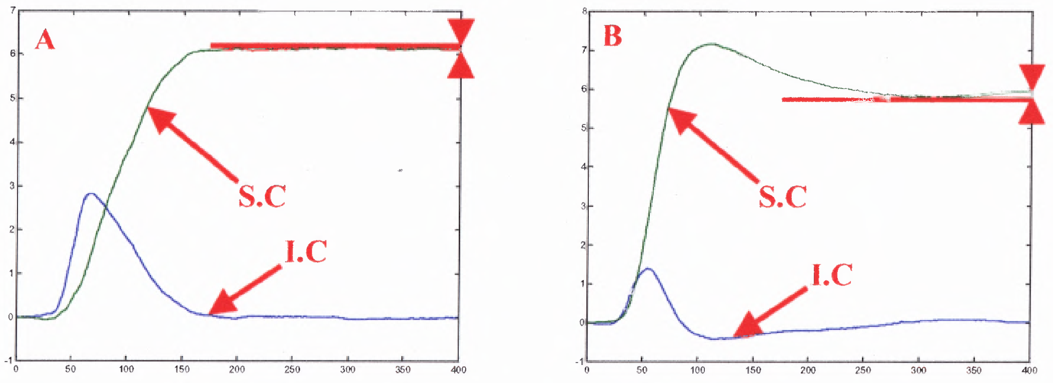

Figure 3.16 Subject M02: (A) Six degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 75 [17]. (B) Six degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 50 [17].

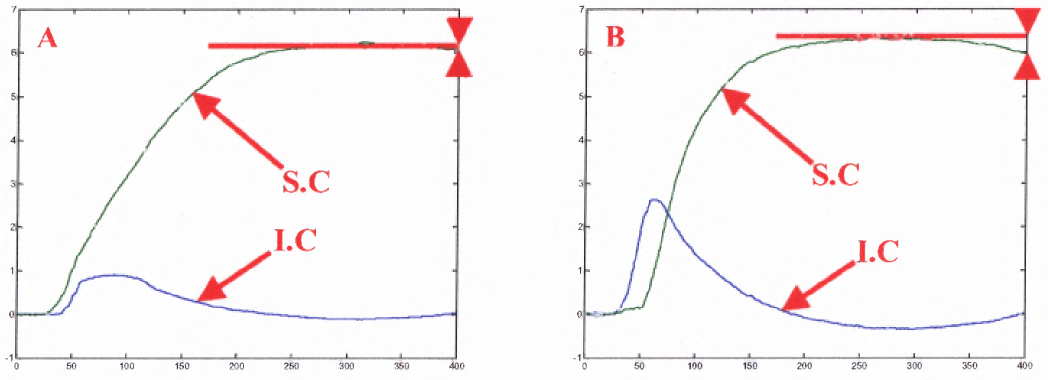

Figure 3.17 Subject F03: (A) Six degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 75 [17]. (B) Six degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 65[17].

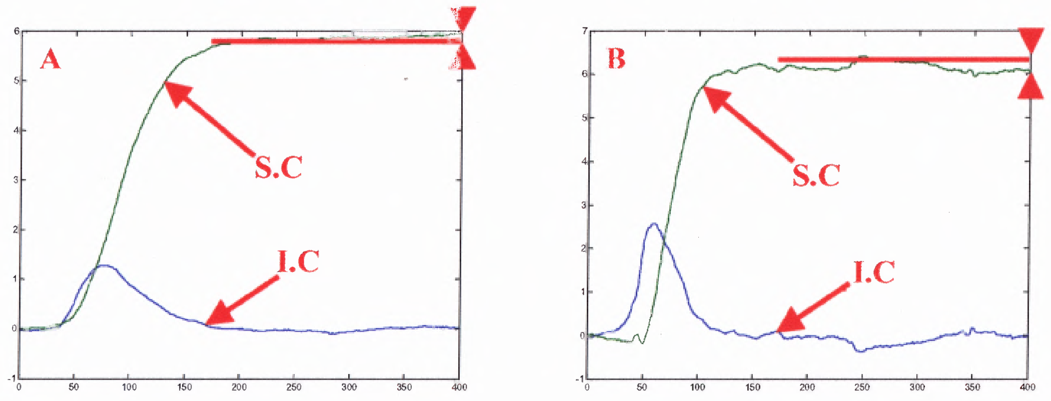

Figure 3.18 Subject D04: (A) Six degree haploscope data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 80 [17]. (B) Six degree LED data. Data truncated from 600 points to 400 points. ICA run to two components. Sustaining component (S.C) and initiating component (I.C) labeled. Partitioned at 70[17].

3.3.3 FastICA to Two Components

ICA to two components on haploscope data (shown in figures 3.19 and 3.20) found the initiating component, and sustaining components of disparity vergence eye movements with constant accommodation. ICA data are plotted in degrees as a function of sample number (time). The components are not to their correct scale. This analysis was used because it allowed ICA to be run to the full length of each trial, 600 points, instead of being limited to 400 points from the icamtl algorithm. For most of the subjects the shapes of the components found with this analysis matched the components found for disparity vergence data with constant accommodation with the icamtl algorithm [15, 17]. The exception can be seen in figure 3.19A where there is a downcurve of both components. The downcurve seen in figure 3.19A can also be seen in the experimental data (figure 3.7A).

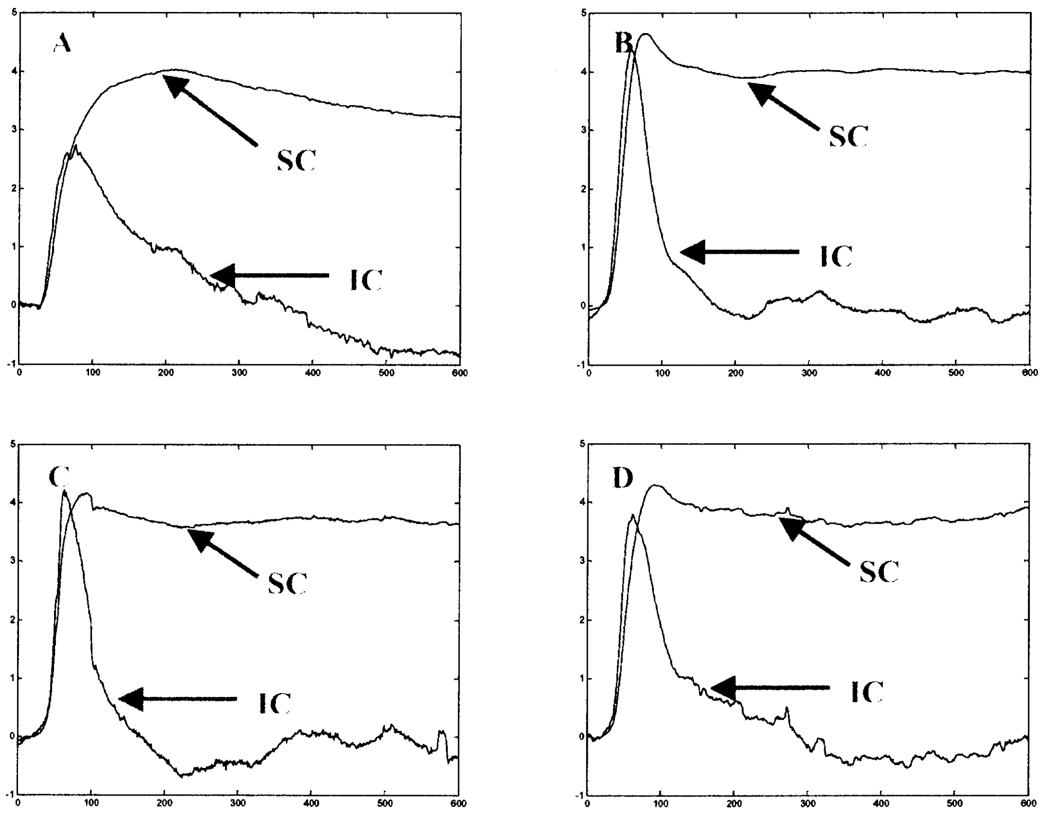

Figure 3.19 ICA on four degree haploscope data to two components. Initiating component (IC) and sustaining component (SC) labeled (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

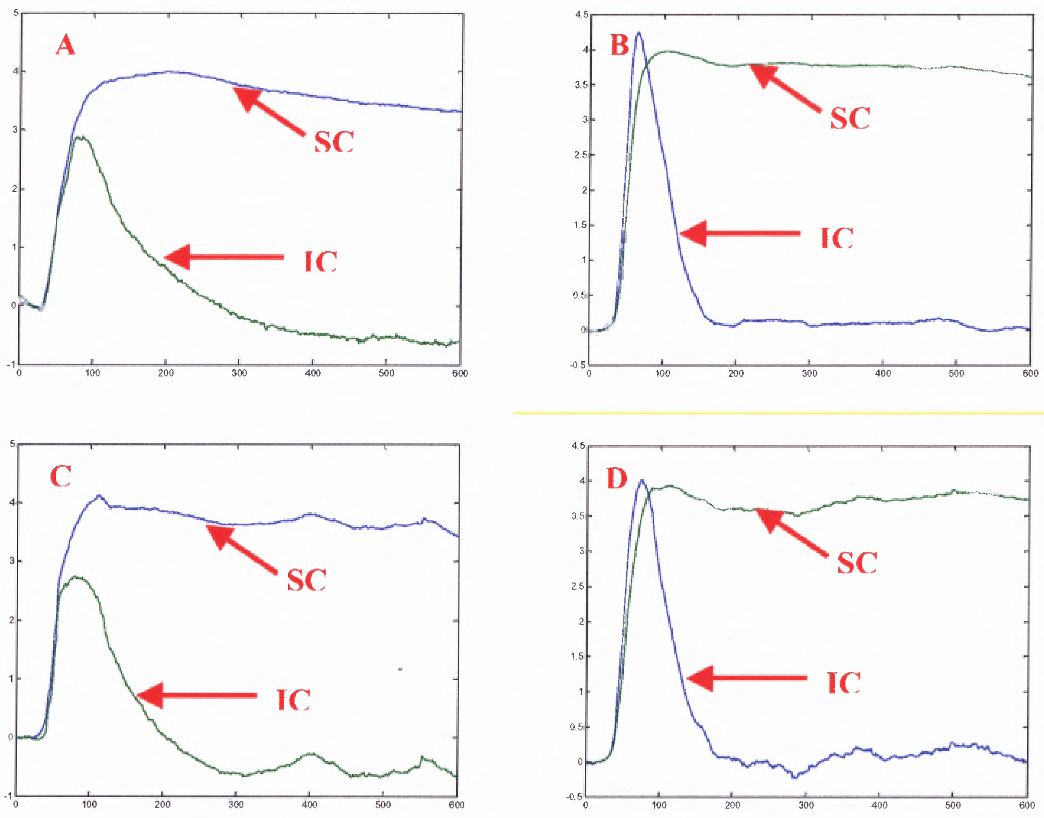

Figure 3.20 ICA on six degree haploscope data to two components. Initiating component (I.C) and sustaining component (S.C) labeled (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

The FastICA algorithm was used to find two components for data collected with disparity vergence with varying accommodation (LED experimentatal setup). There are visible differences in the components found between ICA performed to two components on experimental haploscope, and experimental LED data. These differences in components found further shows that disparity vergence with change in accommodation (LED) is different from disparity vergence with constant accommodation (haploscope). The components are plotted in degrees as a function of sample number (time). The components are not to their correct scale. There is not as visible a patterned difference between the LED data components found in figures 3.21 and 3.22 and the haploscope data components shown in figures 3.19 and 3.20.

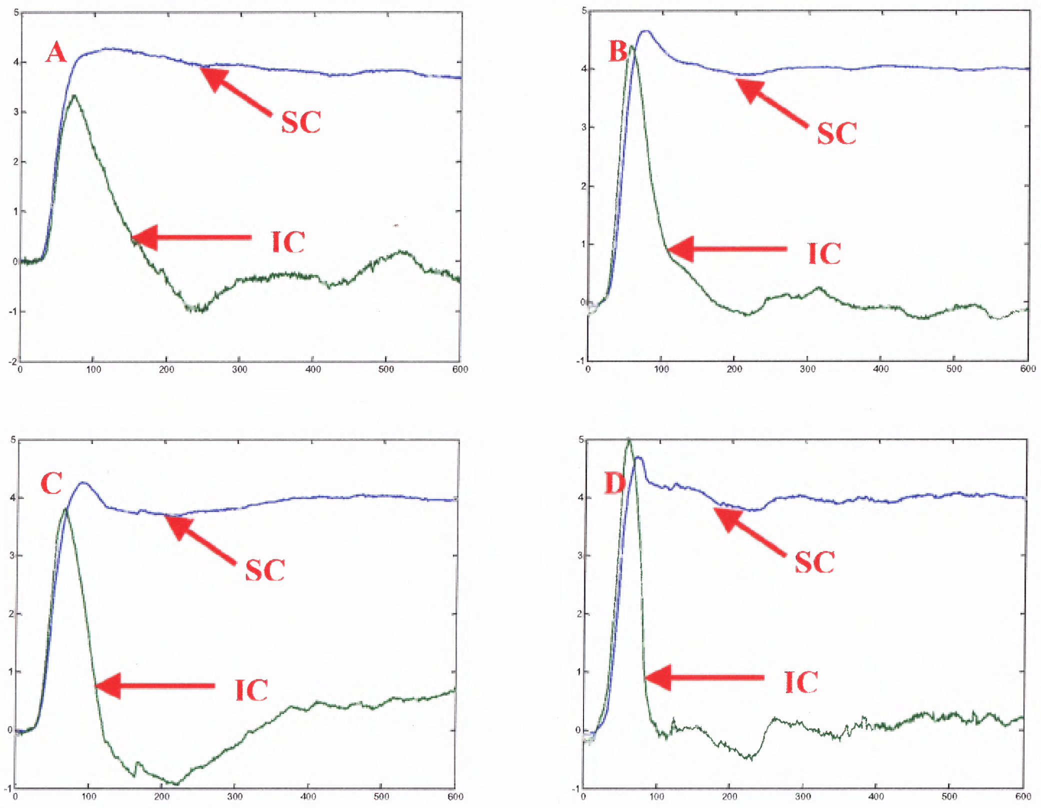

Figure 3.21 ICA on four degree LED data to two components. Initiating component (I.C) and sustaining component (S.C) labeled (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

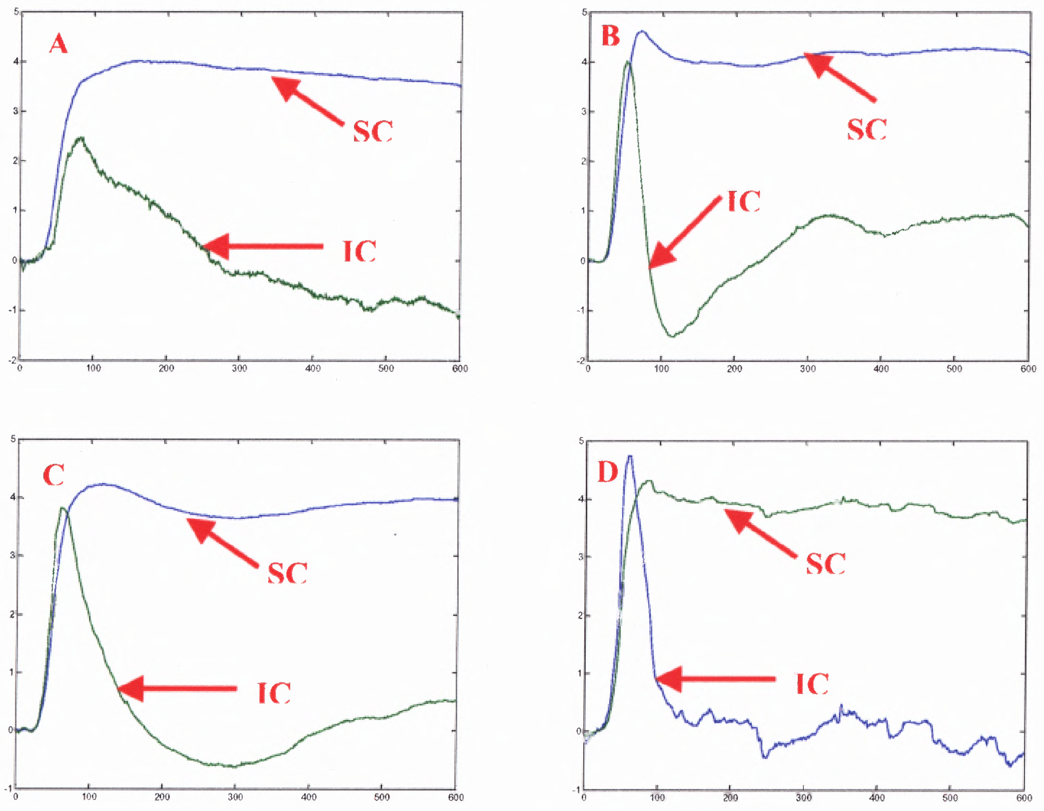

Figure 3.22 ICA on six degree LED data to two components. Initiating component (I.C) and sustaining component (S.C) labeled (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

3.3.4 FastICA to Three Components

FastICA to determine three components was performed on experimental LED data (disparity vergence with varying accommodation). The components are plotted in degrees as a function of sample number (time). The components are not to their correct scale. The third component found was multiplied by negative one if needed to scale this component to match the third component found in the simulated data analysis (figure 2.5C).

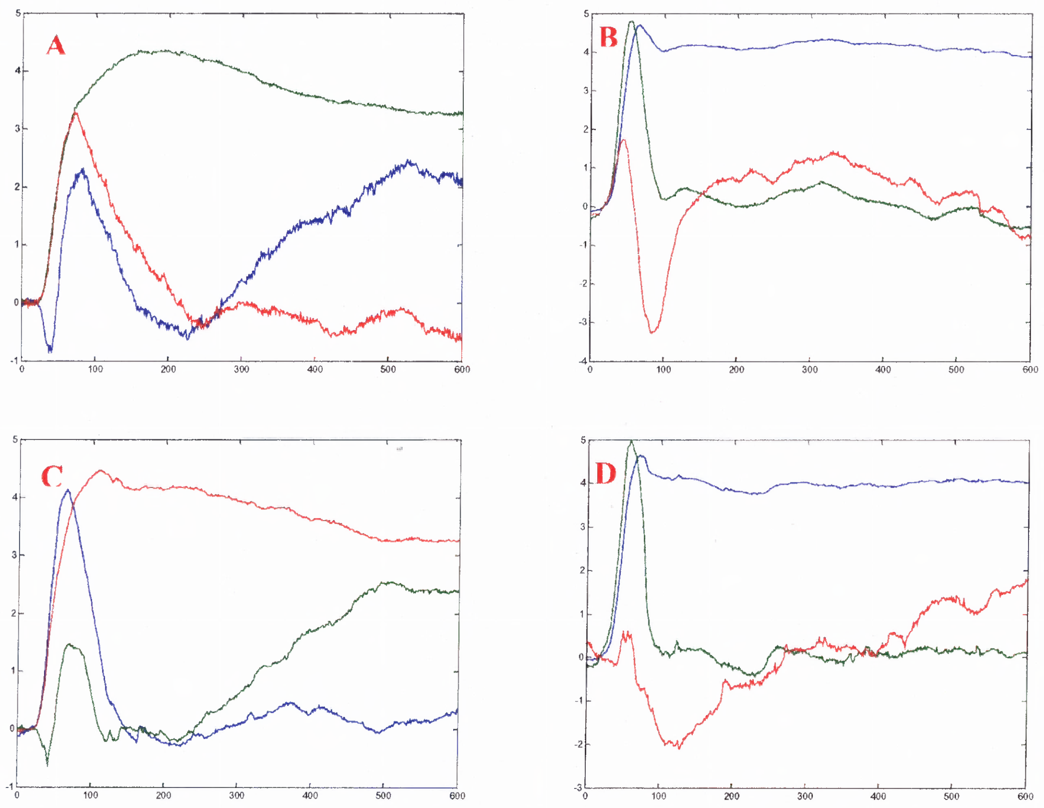

Figure 3.23 ICA on four degree LED data to three components (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

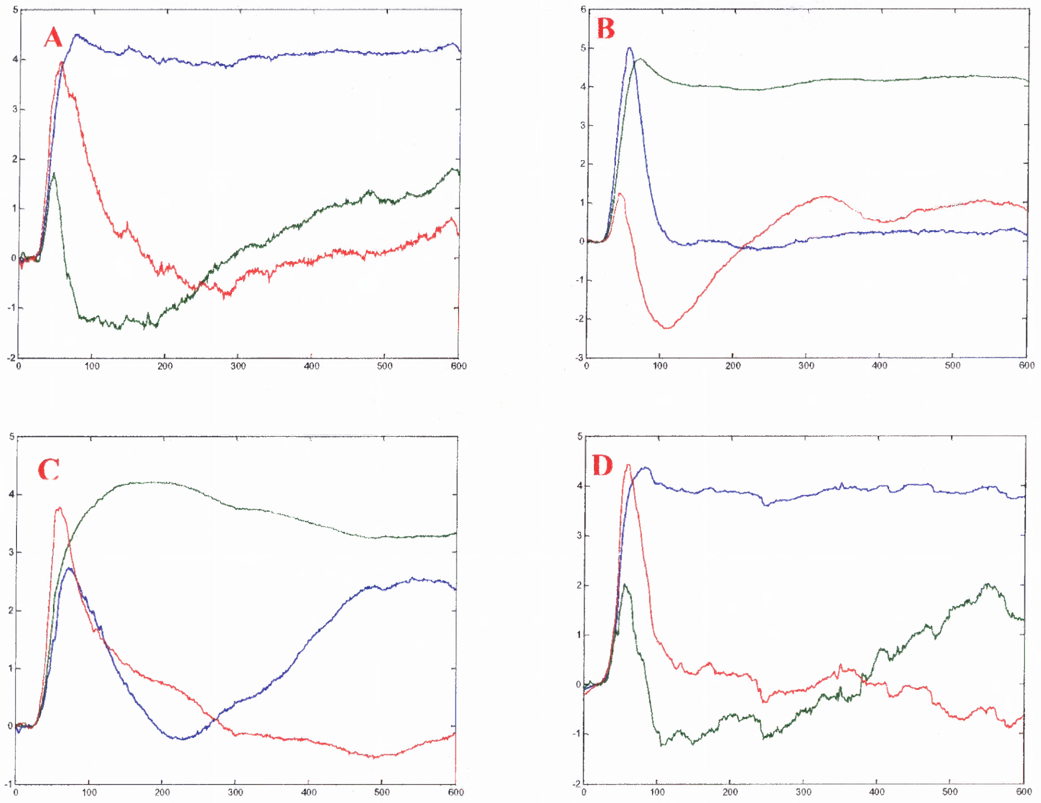

Figure 3.24 ICA on six degree LED data to three components (degrees vs. sample number). (A) Subject B01 (B) Subject M02 (C) Subject F03 (D) Subject D04.

The three components that the independent component analysis (ICA) found (shown in figures 3.23 and 3.24) did not match the three hypothesized components that were input into the simulated data (shown in figure 2.5A). The shapes of the curves found did match the shapes of the curves found in the simulated data analysis (a step component, a pulse component and the 'N' shaped component shown in figure 2.5C). This similarity in curves found adds support to the existence of the hypothesized accommodative component.

CHAPTER 4

DISCUSSION

4.1 Conclusions

The icamtl algorithm written for MATLAB by Dr. Semmlow and his colleagues was used to perform ICA to two components on simulated and truncated experimental data. The successful reproduction of the analysis performed by Semmlow et al confirmed that ICA was working properly [7].

A question that needs to be addressed is the differences in results between the subjects in the icamtl independent component analysis. The main issues are time, step size and perhaps visual acuity of the individual subjects. The six degree LED did not curve down as much as the four degree LED. Because it is a larger step, the accommodative system may have a longer latency compared to a smaller stimulus. The two subjects who did not maintain fixation were both nearsighted. Although the lens kit compensates for subjects' nearsightedness, the targets are still blurry, potentially stimulating the accommodative response earlier.

The FastICA [15] package written for MATLAB was used to perform independent component analysis (ICA) on both experimental and simulated data. When FastICA blindly separated three components on the simulated LED data created for this thesis [16] (figure 2.2B), the resulting components found did not match the three components that were input into the algorithm (figure 2.2A).

The most likely explanation for why the ICA data on the simulated data did not return the components entered is due to loss of independence of the three components.

This loss of independence is probably caused by the similarities in shape between the accommodative component (AC) and the sustaining component (SC). Both the AC and the SC are steps although the accommodative component has a longer latency and is of a smaller magnitude. ICA algorithms do not find the proper scale of the components, it defines the behavior of the components. Principal component analysis (PCA) determines the number of components. ICA will search for the number of components the operator tells the algorithm to search for. While FastICA was able to find a step component that looks like the SC and a pulse component that looks like the IC, it can be observed that the remaining component found is a mixture of the actual components. Thus it is hypothesized that the third component that ICA determines is a combination of the primary and secondary components. This component has an initial pulse shape, and a latter upwards drift giving it an 'N' shape.

When FastICA blindly separated three components on experimental LED data, the resulting components did not match the three hypothesized components. However, there were distinct similarities in the shapes of the components found in the experimental LED data and the simulated LED data. FastICA was able to find a step component that looks like the SC and a pulse component that looks like the IC. It can be observed that the remaining component found is a mixture of a step and a pulse component (figures 3.23 and 3.24). The similarity between components found from the experimental data and components found for the simulated data supports the hypothesized effect of the accommodative component [2, 3, 11, 22].

4.2 Future Research

Initial testing with the FastICA algorithm showed difficulties separating components correctly where two or more of the components were steps (causing loss of independence). Further work needs to be done to improve the algorithm used in order to correctly find the three components of simulated LED data. An additional step that can be implemented to separate the mixed components found in this analysis, is to perform ICA on the remaining components after removing the cleanly isolated step component. This step will probably not be successful though due to the limited number of remaining components. More work also needs to be focused on an algorithm to correctly scale the components found.

ICA performed for this thesis can also be improved by removing instrumentation noise. Ultimately, another blind source separation technique may be needed to find the accommodative component.

Appendix A

IRB Consent Form

A.1 Copy of IRB consent form signed by all subjects who participated in this study. Each subject also received a signed copy of this form for their records.

NJIT Consent Form_CI_ without funding group1 to 3.doc

NEW JERSEY INSTITUTE OF TECHNOLOGY
323 MARTIN LUTHER KING BLVD.
NEWARK, NJ 07102

CONSENT TO PARTICIPATE IN A RESEARCH STUDY
CI Groups 1 to 3 without funding

TITLE OF STUDY: Neural Plasticity Effects of Optometric Vision Therapy for Convergence Insufficiency in the Oculomotor System: An Investigation Utilizing Mathematical Models and Advanced Digital Signal Processing

RESEARCH STUDY: Vergence Oculomotor Control

I,____, have been asked to participate in a research study under the direction of Dr(s). _Tara Alvarez, Michael Lacker, Kenneth Ciuffreda and Barry Tannen__________________________________
Other professional persons who work with them as study staff may assist to act for them.

PURPOSE:

I have been invited to participate in a research study that is designed to study the inward and outward movements by nerves of the eyes. This study will lead to a better understanding of the control of eye movements.

DURATION:

My participation in this study will last for ____approximately 9 months__________.

I will attend a minimum of three sessions over a 9-month period. One session at month 0, a second session at month 3 and a third session at month 9. At times, the researcher may not record enough data on a single one hour experiment and may ask me to return within a few days to finish data collection.

PROCEDURES:

I understand that I will be participating in a study comparing people without binocular problems designated as group 1 to people with the binocular problem known as convergence insufficiency (CI) designated as group 2. I will be placed in one of two groups. If I have symptoms that indicate I may have a high probably of having CI then I will be evaluated by Dr. Ciuffreda to determine if I will part of group 1 or group 2.

I have been told that, during the course of this study, the following will occur:

An eye movement monitor will be placed on my head to objectively record my eye movements. The equipment will not physically touch my eyes. A low intensity infra red light will be shone into my eyes. I will use a chin rest or a bite bar to reduce head movement. I will press a button to signify I am ready to begin an experiment and will try not to blink during the experiment, which lasts five to twenty seconds. I will look at targets that will move some time after the button is pressed.

PARTICIPANTS:

I will be one of about ____50______participants to participate in this trial.

EXCLUSIONS:

I will inform the researcher if any of the following apply to me:

- ❖ During experiments any stress or fatigue is experienced.
- ❖ My eyes feel dry.

RISK/DISCOMFORTS:

I have been told that the study described above may involve the following risks and/or
discomforts:

I may experience fatigue during the experiments and possibly drying of my eyes. If drying of my eyes occurs then I should blink. If I get tired, I can ask for a rest period.

There also may be risks and discomforts that are not yet known.

CONFIDENTIALITY:

Every effort will be made to maintain the confidentiality of my study records. Officials of NJIT will be allowed to inspect sections of my research records related to this study. If the findings from the study are published, I will not be identified by name. My identity will remain confidential unless disclosure is required by law.

PAYMENT FOR PARTICIPATION:

I have been told that I will not receive monetary compensation for my time.

CONSENT AND RELEASE:

I fully recognize that there are risks that I might be exposed to by volunteering in this study which are inherent in participating in any study; I understand that I am not covered by NJIT's insurance policy for any injury or loss I might sustain in the course of participating in the study.

I agree to assume and take on myself all risks and responsibilities in any way associated with this activity. I release NJIT, its trustees, agents, employees and students from any and all liability, claims and actions that may arise as a result of my participation in the study. I understand that this means that I am giving up my right to sue NJIT, its trustees, agents and employees for injuries, damages or losses I may incur.

I have not had exposure to metallic particulate matter and have not participated in laser surgery of any kind on my eyes.

RIGHT TO REFUSE OR WITHDRAW:

I understand that my participation is voluntary and I may refuse to participate, or
may
discontinue my participation at any time with no adverse consequence. I also understand that the investigator has the right to withdraw me from the study at any time.

If it is found that I do have convergence insufficiency, I agree to postpone treatment for nine months (duration of study) OR, if I decide to seek treatment, I would information the research group. In the latter case, the research group may decide to exclude my further participation in the study.

INDIVIDUAL TO CONTACT:

If I have any questions about my treatment or research procedures that I discuss them with the principal investigator. If I have any addition questions about my rights as a
research subject, I may contact:

Richard Greene, Chair of the Institution Review Board at (973) 596-3281.

SIGNATURE OF PARTICIPANT

I have read this entire form, or it has been read to me, and I understand it completely. All of my questions regarding this form or this study have been answered to my complete satisfaction. I agree to participate in this research study.

Subject: Name:____________________
Signature:__________________________

Date:____________________________

SIGNATURE OF READER/TRANSLATOR IF THE PARTICIPANT DOES NOT READ ENGLISH WELL

The person who has signed above,
____, does not read English well, I read English well and am fluent in (name of the language) ____________________________________, a language the subject understands well. I have translated for the subject the entire content of this form. To the best of my knowledge, the participant understands the content of this form and has had an opportunity to ask questions regarding the consent form and the study, and these questions have been answered to the complete satisfaction of the participant (his/her parent/legal guardian).

Reader/
Translator: Name:______________________
Signature:_________________________
Date:_______________________________

SIGNATURE OF INVESTIGATOR OR RESPONSIBLE INDIVIDUAL

To the best of my knowledge, the participant, ____, has understood the entire content of the above consent form, and comprehends the study. The participants and those of his/her parent/legal guardian have been accurately answered to his/her/their complete satisfaction.

Investigator's Name:______________________
Signature:_______________________

Date:____________________________________

Appendix B

MATLAB Scripts

Matlab scripts that were edited/written for this thesis. All were written/run under MATLAB 6.5 unless otherwise stated.

B.1 visresc2call.m

Used to categorize experimental data. Modified to save vergence data to have ICA performed on. Run under MATLAB 6.1.

```
%Program eom
%   NJIT Vision Research Analysis
%????????????????????????????????????????????????????????????????????????
%
%          Control File Parameters:
%               1) Simulation flag (0 = no)
%               2) Data length (in sec)
%               3) End of error calculation [optional]
%               4) Stimulus amplitude (in deg) [optional]
%
%?????????????????????????????????????????????????????????????????????????
?
%  Flags:
%      sim_flag:       y = simulate this pass;
%                      t = disable simulation this pass only, but plot
%                      n = simulation disabled always
%               d = disable simulation and plotting this pass only
%               e = disable simulation always and plotting this pass only
%               X = disable simulation and plotting always
%
%      param_flag:     i = no param file, use initial conditions (first recd only)
%                      s = no param file, use existing values (from last record)
%                      y = valid param file, param taken from param file
%                      n = valid param file, but not for this record
%--------------------------------------------------------------------------

global cntl_flag converg_flag sim_flag opt_flag optional_param nurec x err1;
global isp istart comments legend response y t deriv_sim sim_flag deriv test_response;
global Dum D_index scale Dur T rec_nu  iskip i_index stim_p stim;
global fig1 fig2 fig3 fname diff_save_flag max_left max_right max_time_left
max_time_right;
global cntl_p first_flag l_cal r_cal max_i_vel n_avg tau1 tau2;
```

```
close; close;close;              % Close any open figures from previous run
fclose all;
clear param;

%   -----Initialize parameters-----

cntl_flag = 'p';  % Controls plotting
nu_param = 12;          % Number of parameters
rec_length = 600;       % Data record length  (default for 3 second data)
T = 1/200;            % Sample frequency
print_flag = 'i';       % Indicates no figures have yet been printed (changes to x)
diff_save_flag = 'i';   % Indicates no diff curves have been saved yet.
opt_flag = 'n';        % Incidatest that optomization is off
cmd = 'a';            % Contols options (default a = advance)
iskip = 8;            % Derivatve skip factor
iskip1 = 4;                % Acceleration skip factor
scale = (2*iskip+1) * T;  % Derivative scale factor
scale1 = (2*iskip1+1) * T; % Acceleration scale factor
first_flag = 'y';      % Indicates first time for special plot
sim_flag = 'n';        % Simulation flag.  See above.
cal_flag = 2;              % Indicates calibration method,; default is 2 pts
flag_tau_study = 1;       % ==1 indicates tau program on
error = [];
stw_mech = [];
phase_flag = 1;           % indicates first time through phase loop
categorization_flag = 1; % indicates first time through categorization loop
%   -----Set up directories-----

export_dir = 'c:\axumw\data\';                % Default directory for data export
param_dir  =  'c:\matlab6p1\usr\data\ic_param';          % Default directory for model
parameters
eom_data_dir = 'c:\vision_eye_movement_data\'; % Default directory for eye movement
data
eom_cata_dir = 'c:\vision_eye_movement_data\cata\'; % directory for categoried eye data
eom_ica_dir  =  'c:\vision_eye_movement_data\ica\';  %  directory  for  eye  movements
categorized for ica analysis
iso_data_dir  =  'c:\matlat6p1\usr\data\iso_data';  %  Default  directory  for  isolated  slow
components


%   -----Set up figures-----

close; close;
fig1 = figure('Units','inches','Position',[0 4 3.5 3.5]);
orient portrait;
```

```
fig2 = figure('Units', 'inches','Position',[3.6 4 3.5 3.5]);
orient portrait;

fig3 = figure ('Units', 'inches', 'Position', [7.2 4 3.5 3.5]);
orient portrait;
fig4 = figure ('Units', 'inches', 'Position', [7.2 4 3.5 3.5]);
orient portrait;
fig5 = figure ('Units', 'inches', 'Position', [7.2 4 3.5 3.5]);
%___________ Set up Digital Filters________________________

[posb posa] = butter(8,.2);  % Position filter % tried increasing filter from 0.2
[velb vela] = butter(8,.5);  % Velocity filter

%    -----Get figure legend and data file name-----

legend = input('Input running title: ','s');


%    -----Get control file and set appropriate parameters--------

cd c:\matlab6p1\usr\verg_mod;

if exist('cntl_p.d') == 2
    load 'cntl_p.d';
else
    disp('Control file missing');
    return;
end % if exist('cntlp.d') ==2

if cntl_p(1) < 0   % Abort simulations
        sim_flag = 'n';     % Plot only flag
else
        sim_flag = 'y';
end % if cntl_p(1) <0
% this section is commented out by Tara will have t = to the time of experiment
%if cntl_p(2) == 1  % Indicates 1 sec data
%       rec_length = 200;
%       t = (0:T:.999)';
%elseif cntl_p(2) == 2   % Indicates 2 sec data
%       rec_length = 400;
%       t = (0:T:1.999)';
%elseif cntl_p(2) == 3   % Indicates 3 second data
%       rec_length = 600;
%       t = (0:T:2.999)';
%end %cntl_p(2) ==1
```

```
if length(cntl_p) > 2
        isp = control_param(3);
end %length(cntl_p) >2

%   -----Zero working arrays-----

x1 = zeros(1,nu_param+5);
deriv_sim = zeros(rec_length,2);
response = zeros(rec_length,2);
avg = zeros(rec_length,1);
sum_sq = zeros(rec_length,1);
der_avg = zeros(rec_length,1);
sum_sq = zeros(rec_length,1);
sum_sq_der = zeros(rec_length,1);
stdm = zeros(rec_length,1);
std_dev = zeros(rec_length,1);

%    -----Get data and parameter files-----
which_dir = input('Enter file type (cat or dat or asc)', 's');
name = input('Enter file name (6 char)', 's');
comments = name;
if strcmp (which_dir , 'dat')==1;
        fname = [eom_data_dir, name, '.dat'];
        asc_flag = 'n'; %can do simulation data on individual records
else
        if strcmp (which_dir , 'cat') ==1;
                fname = [eom_cata_dir, name, '.cat'];
                asc_flag = 'n';%can do simulation data on individual records
        else
                if strcmp(which_dir, 'asc')==1;
                        fname = [export_dir, name, '.asc'];
                        asc_flag = 'y';%do simulation data on average data
                end % strcmp(which_dir, 'asc') ==1;
        end % strcmp(which_dir, 'cat') ==1
end % strcmp(which_dir, 'dat') ==1
if exist(fname) == 2
        fid = fopen(fname);
else
        disp('Error: data file missing');
        return;
end %exist(fname) ==2
if sim_flag == 'n'
```

```
        param_flag = 'n';
end
if sim_flag == 'y'
        fname_prm  = [param_dir fname '.prm'];              %   Change   director   to   IC
model parameter,
        if exist([fname_prm]) == 2              %   check if exists, then load parameters
                eval(['load ', fname_prm, ' -ascii']);
                eval(['param = ', fname;]);
                eval(['clear ',fname]);
                [nu_param_rec, npar] = size(param);  % Check if parameter file intact
                if npar == nu_param +1 & nu_param_rec >= 1
                        param_flag = 'y';
                else
                        param_flag = 'i';          % Parameter file not valid, use initial values
                        disp ('Parameter size error - number param; number recds: ');
                        disp([npar, nu_param_rec]);
                        clear param; % Clear so new param array can be created

                end %npar == nu_param +1
        else

                param_flag = 'i';          % Also use initial values if parameter file not found
                disp('Parameter error:  file not found');
        end %exist([fname_prm]) ==2

end % sim_flag == 'y'


disp('Search for the following stimulus');
desired_stim = input('Enter [stim type, l amp, r amp, l bias, r bias, lgain, r gain]');
if desired_stim == []
        all_stim_flag = 'y';
else
        all_stim_flag = 'n';
        desired_stim = desired_stim';
end %desired_sim == []

% --------Initialize Variables ----------------------------------

rec_count = 0;
rec_nu = 0;
next_rec = 'y';
n_avg = 0;
if asc_flag == 'y'
                data = fscanf(fid, '%f', [6,600]); % Tara just for now
                data = data';
                %t = data(:,1);
                vergence = data(:,2);
                deriv = data(:,3);
                clear data;

        end

%-------------------------- Main Loop ------------------------
while(1)  % Loop continuously

 if asc_flag =='n';
  if next_rec == 'y'              % Do this section on time only for each new record
        if sim_flag == 't'        % Reset sim_flag
                sim_flag = 'y';  % Do simulations
        end % sim_flag == 't'
        next_rec = 'n';           % Set next record flag to 'n'
        stim1 = zeros(7,1);
        stim = zeros(7,1);
        while (max(stim1 ~= desired_stim))    % Search for next desired stimulus
                %  -----Check for end of file -----
                [file_name count] = fscanf(fid, '%s', 1);
                if count == 0    % If end of file check for new file to analyze
                 new_name = input('Record complete, enter new file name to continue:
','s');

                 if isempty(new_name) == 0
                        comments = new_name;
                        fname = [eom_data_dir, new_name, '.dat'];
                        if exist(fname) == 2
                                fid = fopen(fname);
                        else
                                disp('Error: data file missing');
                                return;
                        end % exist(fname ==2)
                        file_name = fscanf(fid, '%s',1);
                 else
                        disp(' Record complete, saving data');
                        if exist('file_out') ~= 0


                         avg  = [t avg/n_avg der_avg/n_avg stdm std_der acceleration];
                         eval(['save ', file_out, '.asc avg -ascii;']);
                        end %exist('file_out') !=0
                        if exist('file_out1') ~= 0
                         eval(['save ', file_out1, '.asc stw_mech -ascii;']);
                        end %exist('file_out') !=0
                        fclose all;
                        return;
                  end %isempty(new_name) == 0
                end %count == 0
                file_date = fscanf(fid, '%s', 1);
                total_pts = fscanf(fid, '%d',1);
        t = (0:T:(total_pts-1)*T); % t will be in seconds
                nu_cal_pts = fscanf(fid, '%d',1);
                stim = fscanf(fid,'%f',7);

                if all_stim_flag == 'y'
                        stim1 = desired_stim;  % Force stimulus match
                else
                        stim1 = stim;
                end %all_stim_flag == 'y'

                if stim(2) > 0    % Test for stimulus direction
                        converg_flag = 'y';
                else
                        converg_flag = 'n';
                end %stim(2) > 0


                delay = fscanf(fid, '%f',1);

                cal_type = fscanf(fid, '%s',1);
                dummy = fscanf(fid, '%d',4);


                l_cal = fscanf(fid, '%f', [2, nu_cal_pts]);

        l_error = fscanf(fid, '%f',nu_cal_pts);

                l_2_cal = fscanf(fid, '%f', [2,2]); % changed from Rutgers %d to %f
                r_cal = fscanf(fid, '%f', [2,nu_cal_pts]);
                r_error = fscanf(fid, '%f',nu_cal_pts);
                r_2_cal = fscanf(fid, '%f', [2,2]); % changed from Rutgers %d to %f
                response = fscanf(fid, '%f', [total_pts,2]);  % changed from Rutgers from
    %d to %f
                rec_count = rec_count + 1;



        end %while (max(stim1 !=desired_stim))
```

```
		response = response(1:total_pts,:);
		stim_p = stim(2);

	% Save 3 point calibration data

	if cal_flag > 2
	 l_linearity  =  (((l_cal(3,2)  +  l_cal(1,2))/2  -  l_cal(2,2))/(l_cal(3,2)-l_cal(1,2)))  *
100;
	 r_linearity  =  (((r_cal(3,2)  +  r_cal(1,2))/2  -  r_cal(2,2))/(r_cal(3,2)-r_cal(1,2)))  *
100;
	 comments = ['Linearity (l/r):  ', num2str(l_linearity),';   ', num2str(r_linearity)];
	else
	 l_linearity = 0;
	 r_linearity = 0;
	end %cal_flag > 2



	%   Check for overflow or underflow on both channels
	[m i] = max(response(:,1));
	if m > 4095
		disp('ERROR: left eye overflow at sec:');
		disp(i*T);
	end %m > 4095
	[m i] = max(response(:,2));
	if m > 4095
		disp('ERROR: right eye overflow at sec:');
		disp(i*T);
	end %m > 4095
	[m i] = min(response(:,1));
	if m < 1
		disp('ERROR: left eye underflow at sec:')
		disp(i*T);
	end %m < 1
	[m i] = min(response(:,2));
	if m < 1
		disp('ERROR: right eye underflow at sec:');
		disp(i*T);
	end %m < 1

	if cal_flag == 3
		cal_3m;   % Calibrate response
	else
```

```
%%%%%2                                                              point
Calibration%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%55
    l_cal_low = 0;
    for (i = 1:10)
      l_cal_low = l_cal_low + response(i,1);
    end
    l_cal_low = l_cal_low / 10.0;


    l_cal_high = 0;
    for (i = (total_pts-99):total_pts)
      l_cal_high = l_cal_high + response(i,1);
    end
    l_cal_high = l_cal_high/100.0;
    % changes


    %l_stim_low = l_2_cal(1,1);
    %l_stim_high = l_2_cal(2,1);
    l_stim_low = stim(4);
    l_stim_high = stim(4) +stim(2);
    r_cal_low = 0;
    for (i = 1:10)
      r_cal_low = r_cal_low + response(i,2);
    end
    r_cal_low = r_cal_low / 10.0;


    r_cal_high = 0;
    for (i = (total_pts-99):total_pts)
      r_cal_high = r_cal_high + response(i,2);
    end
    r_cal_high = r_cal_high/100.0;
    %r_stim_low = r_2_cal(1,1);
    %r_stim_high = r_2_cal(2,1);
    r_stim_low = stim(5);
    r_stim_high = stim(5)+stim(3);
    if l_stim_high == l_stim_low | l_cal_high == l_cal_low
            disp('ERROR Left Calibration');
            l_slope = 1;
    else
            l_slope = (l_cal_high - l_cal_low)/(l_stim_high - l_stim_low);
    end
    l_b = l_cal_low - l_slope * l_stim_low;

    if r_stim_high == r_stim_low | r_cal_high == r_cal_low
```

```
                disp('ERROR Right Calibration');
                r_slope = 1;
        else
                r_slope = (r_cal_high - r_cal_low)/(r_stim_high - r_stim_low);
        end
        r_b = r_cal_low - r_slope * r_stim_low;

        for i = 1:total_pts
                response(i,1) = (response(i,1) - l_b)/l_slope;
                response(i,2) = (response(i,2) - r_b)/r_slope;
        end

%%%%%%%%%%%%%%%%%% end 2 point calibration



            end %cal_flag == 3


            % Subtract baseline
            response(:,1) = response(:,1) - mean(response(13:24,1));
            response(:,2) = response(:,2) - mean(response(13:24,2));

            rec_nu = rec_nu + 1;

% ---------Apply Digital Position filter --------------------
            %response(:,1) = filtfilt(posb, posa, response(:,1)); % commented back in by Tara
to apply filters

  % response(:,2) = filtfilt(posb, posa, response(:,2)); % commented back in by Tara to
apply filters
            deriv = zeros(length(response(:,1)),2);
            accel = zeros(length(response(:,1)),2);
            for id = iskip+1:length(response(:,1))-iskip
                    deriv(id,1) = (response(id+iskip,1) - response(id-iskip,1)) / scale;
                    deriv(id,2) = (response(id+iskip,2) - response(id-iskip,2)) / scale;
            end % id = iskip+1:length(response(:,1))-iskip

            %calculate acceleration
            for id = iskip + iskip1 + 1:length(deriv(:,1))-iskip1
                    accel(id,1) = (deriv(id+iskip1,1) - deriv(id-iskip1,1)) / scale1;
                    accel(id,2) = (deriv(id+iskip1,2) - deriv(id-iskip1,2)) / scale1;
            end % id = iskip+1:length(response(:,1))-iskip

% ----------Apply Digital Velocity Filter -----------------------------
```

```
        %deriv(:,1) = filtfilt(velb, vela, deriv(:,1));
        %deriv(:,2) = filtfilt(velb, vela, deriv(:,2));

        [max_left, max_i_left] = max(deriv(:,1));
        [max_right, max_i_right] = max(deriv(:,2));
        [max_verg_vel, max_i_vel] = max((deriv(:,1) + deriv(:,2))/2);
        avg_max_vel = (max_left + max_right)/2;
        max_time_left = max_i_left * T;
        max_time_right = max_i_right * T;

  end %next_rec == 'y'
end %sim_flag =='n'


        %  Assign initial parameters only if not previously assigned
        %       Default parameters

        if param_flag == 'i'
                x(1) = .13;          % Delay
                x(2) = 19;           % Slew Rate
                x(3) = .8;          % Step Gain
                x(4) = .2;          % Pulse Gain
                x(5) = 1.2;          % Local Fbk
                x(6) = 1.;          % IC Gain
                x(7) = 6.7;          % SC Gain
                x(8) = .55;          % SC Deriv gain
                x(9) = .14;          % Left Plan tau
                x(10) = 12.0          % SC Deiv tau
                x(11) = .235            % SC onset delay
                x(12) = 9.0;          % Right Plant tau

                param_flag = 's';  % Parameter set, do not reset
        elseif param_flag =='y'

            for param_pos = 1:nu_param_rec
                if(param(param_pos,1)) == rec_nu % Search for appropriate param
                    x = param(param_pos,2:npar);
                    param_flag == 'y';  % Valid parameters found
                else
                    param_flag = 'n';
                end %(param(param_pos,1)) == rec_nu
            end %param_pos = 1:nu_param_rec
        end % param_flag == 'i'
        if param_flag ~= 'y'
                param_pos = rec_nu;
        end %param_flag ~= 'y'


%     ------ Simulate and/or plot response------

        if sim_flag == 'y';          % Simulate and plot
                vergmodt;

        elseif sim_flag == 't'     % Plot only, but restore simulation for next time

                plot_icm;
                sim_flag = 'y';  % Turn simulation back on
        elseif sim_flag == 'd'     % No plot, turn simulation back on
                sim_flag = 'y';
        elseif sim_flag == 'n'

                plot_icm;         % Plot only, do not restore simulation flag
        elseif sim_flag == 'e'     % No plot, no simulation
                sim_flag = 'n'
        end % sim_flag == 'y';

%     ----- Request  Command-----
        disp('x(accept) a(dv) g(n chg) n(param) p(rint) m(ult plt) s(save sim)
c(categorize)')
        cmd = input(' d(iff plt) t(ext) q(uit) k(bd) e(xpt) y(dynam) h(phase pln) w(double
pln)','s');

%     ------Execute specific Command-----------

% ------------Categorization of Data: allows user to save only files that do not contain
artifacts to a data file
  % puts data into raw data forwat and attachs a .cat suffix also creates an ICA matrix
and attaches a .ica suffix

  if cmd == 'c'
   if categorization_flag == 1 % first time through categorization loop

    categorize_outputfilename = input('What is the output file name for categorized
data: ','s');

    file_out = [eom_cata_dir, categorize_outputfilename, '.cat'];
                vergence_ica_matrix_name = 'vergence_ica_matrix';
        fid1 = fopen(file_out,'a'); % can create a path with fopen

         fprintf(fid1,'%s \n%s \n%d \n%d \n%f \n%f \n%f \n%f \n%f \n%f
\n%f\n',file_name, file_date, total_pts, nu_cal_pts, stim(1), stim(2), stim(3), stim(4),
stim(5), stim(6), stim(7));
         fprintf(fid1,'%f\n%s\n%d\n%d\n%d\n%d\n',          delay,          cal_type,
dummy(1),dummy(2),dummy(3),dummy(4));

         fprintf(fid1,'%d\n',l_cal);
         fprintf(fid1,'%f\n%f\n',l_error,l_2_cal);
         fprintf(fid1,'%d\n',r_cal);
         fprintf(fid1,'%f\n%f\n',r_error,r_2_cal);
         fprintf(fid1,'%f\n%f\n',response(1:total_pts,1), response(1:total_pts,2));

       fclose(fid1);
       file_out_ica = [eom_ica_dir, categorize_outputfilename, '.mat'];
      % fid2 = fopen(file_out_ica,'a'); % can create a path with fopen

       vergence = [response(1:total_pts,1)+ response(1:total_pts,2)];
       vergence_ica_matrix = [vergence];

       save (file_out_ica,vergence_ica_matrix_name);


       fid2 = fopen(file_out_ica,'a'); % can create a path with fopen

       vergence = [response(1:total_pts,1)+ response(1:total_pts,2)];
       vergence_ica_matrix = [vergence];

       fprintf(fid2,'%f\n',vergence_ica_matrix);
      fclose(fid2);

     categorization_flag = 2
    else % second or more times through categorization loop

       fid1 = fopen(file_out,'a'); % can create a path with fopen

        fprintf(fid1,'%s \n%s \n%d \n%d \n%f \n%f \n%f \n%f \n%f \n%f
\n%f\n',file_name, file_date, total_pts, nu_cal_pts, stim(1), stim(2), stim(3), stim(4),
stim(5), stim(6), stim(7));
        fprintf(fid1,'%f\n%s\n%d\n%d\n%d\n%d\n',          delay,          cal_type,
dummy(1),dummy(2),dummy(3),dummy(4));
        fprintf(fid1,'%d\n',l_cal);
        fprintf(fid1,'%f\n%f\n',l_error,l_2_cal);
        fprintf(fid1,'%d\n',r_cal);
        fprintf(fid1,'%f\n%f\n',r_error,r_2_cal);
        fprintf(fid1,'%f\n%f\n',response(1:total_pts,1), response(1:total_pts,2));

       fclose(fid1);

            file_out_ica = [eom_ica_dir, categorize_outputfilename, '.mat'];

        %fid2 = fopen(file_out_ica,'a'); % can create a path with fopen
        vergence = [response(1:total_pts,1)+ response(1:total_pts,2)];
        vergence_ica_matrix = [vergence_ica_matrix vergence];
        save (file_out_ica,vergence_ica_matrix_name);

        %fprintf(fid2,'%f\n',vergence_ica_matrix);
      %fclose(fid2);


    end % if categorization_flag ==

   end   %if cmd == 'c'

%  ----------Phase plane plot ---------------
        if cmd == 'h'

         figure(fig3);   % Use figure 3 for this plot
         clf;
         %tara axis([-2 6 -10 30]);
         verg = response(:,1) + response(:,2);
    d_verg = (deriv(:,1) + deriv(:,2));
         if stim(2) <0
        verg = - verg; % used to seeing positive phase domains will switch back to neg at
file saving
        d_verg = -d_verg; % used to seeing positive phase domains will switch back to
neg at file saving
    end
    plot(verg(1:(total_pts- 50)), d_verg(1:(total_pts - 50)))   % Phase plane plot
        %changed to only show first 2 seconds of data

         xlabel('Vergence (deg)');
         ylabel('Velocity (deg/sec)');
         if (input('Accept this data: (y or n) ','s') == 'y')
              disp('*******************        click       on        max        position
*******************');
              [pos, vel,button] = ginput(1);
              [max_d_verg, max_index] = max(d_verg(1:100));
              [min_d_verg, min_index] = min(d_verg);
        time_max_dverg = max_index/600;
        time_min_dverg = min_index/600;
```

```
                top_slew = verg(max_index);
                slew = top_slew/pos;

      if stim(2) <0
          max_d_verg = - max_d_verg; % used to seeing positive phase domains will
switch back to neg at file saving
          pos = - pos; % used to seeing positive phase domains will switch back to neg at
file saving
     end

                title([comments, ' Pos: ', num2str(pos),'  Recd#:', num2str(rec_nu) ] );
          hold on;
                figure(fig4);
                hold on;
                plot(verg(1:275), d_verg(1:275))   % Phase plane plot

                xlabel('Vergence (deg)');
                ylabel('Velocity (deg/sec)');
                title([comments,  '  Recd#:', num2str(rec_nu) ] );
                % tara axis([-3 6 -20 10]);
                if phase_flag == 1
                     h_x_out = [ rec_nu max_d_verg pos max_d_verg/pos min_d_verg
time_max_dverg time_min_dverg max_d_verg/pos slew min_d_verg/max_d_verg];
                     out_name = input('Input output file for v_max and x_max: ','s');
                     file_out = [export_dir, out_name];
                     eval(['save ', file_out, '.txt h_x_out -ascii;']);

                     phase_flag=2;
                else

                     h_x_out = [h_x_out; rec_nu max_d_verg  pos max_d_verg/pos
min_d_verg    time_max_dverg    time_min_dverg    max_d_verg/pos    slew
min_d_verg/max_d_verg];
                     eval(['save ', file_out, '.txt h_x_out -ascii;']);
                end %phaseflag
          end %if statement
     end %'h'

%  ----------Double Phase plane plot ---------------
          if cmd == 'w'
            if(flag_tau_study == 1) % 1 = on

                tau1 = input('Input tau1:  ');
                tau2 = input('Input tau2:  ');
                amp = 4;
```

```
                    stimulat_dat = amp*(1 + ((tau1/(tau2 - tau1))*exp(-t/tau1)) - ((tau2/(tau2 -
tau1))* exp(-t/tau2)));
                    deriv_stimulat_dat = amp*( (( -1/(tau2 - tau1))*exp(-t/tau1)) + ((1/(tau2 -
tau1))*exp(-t/tau2)));
                    figure(fig1);
                    clf;
                    hold on;
                    axis([-.2 3 -1 30]);
                    plot(t,stimulat_dat);
                    plot(t,deriv_stimulat_dat,'r');
                    figure(fig3);
                    plot(stimulat_dat, deriv_stimulat_dat);

                    verg = stimulat_dat;
                    d_verg = deriv_stimulat_dat;
                    multcomp;
            else
                    multcomp;
            end

        end



%   ------ Define dynamic boundry ------
        if cmd == 'y'
                    d_sum = deriv(:,1) + deriv(:,2);
                    vergence = response(:,1) + response(:,2);
                    figure(fig2);
                    clf; plot(t(1:100), vergence(1:100));
                    hold on;
                    title('Place pointer at movement onset and click');
                    [x,y] = ginput(1);
                    lat = x(1); plot([x(1) x(1)], [0 max(vergence(1:200))],':');
                    text(x(1), 0,['lat = ',num2str(lat),' sec']);
                    figure(fig1);
                    clf; plot(t(1:200),d_sum(1:200));
                    title('Place pointer on max vel. and click');
                    [x,y] = ginput(1);
                    max_i_vel = x(1)/T;
                    max_verg_vel = d_sum(max_i_vel);
                    text(0,16,'Place pointer on end points and click');
                    [x,y] = ginput(2);
                    [onset, slope1, slope2] = lin2_fit(d_sum,min(x)/T,max(x)/T);
                    figure(fig2); clf;
                    x = 1:1:400;   % Plot against number
            plot(x,vergence(1:400));
            hold on;
            plot(x,d_sum(1:400)* .2,'r--');

            x = [onset onset]; y = [0 max(vergence(1:200))];
            plot(x,y,':');
            text(onset, 0,['st = ',num2str(onset*T),' sec']);
            plot([lat/T lat/T], y,':');
            if (input('Accept data (y,n)','s')) == 'y';
             err1 = vergence(onset);
             err2 = vergence(max_i_vel);
             if isempty(stw_mech)
                    stw_mech                    =                     [lat,
onset*T,d_sum(onset),max_verg_vel,max_i_vel*T,err1,err2, slope1/T, slope2/T];
                    out_name = input('Input output file for switching info: ','s');
                    file_out1 = [export_dir, out_name];
             else
                    stw_mech          =          [stw_mech;          lat,
onset*T,d_sum(onset),max_verg_vel,max_i_vel*T,err1,err2, slope1/T,slope2/T];
             end %isempty(stw_mech)
            end %(input('Accept data (y,n)','s')) == 'y';
    end %cmd == 'y'


% ----------- Gain Change -----------------
    if cmd == 'g'    % Scale left or right eye response to compensate for gain error
            rel_gain = input('Input relative l/r channel gain (neg for rt eye)')
            if rel_gain < 0
                    response(:,2) = response(:,2) * abs(rel_gain);
            else
                    response(:,1) = response(:,1) * rel_gain
            end %rel_gain < 0
            deriv = zeros(length(response),2);   % Recompute derivatives
            for id = iskip+1:length(response(:,1))-iskip
                    deriv(id,1) = (response(id+iskip,1) - response(id-iskip,1))/scale;
                    deriv(id,2) = (response(id+iskip,2) - response(id-iskip,2))/scale;
            end %id = iskip+1:length(response(:,1))-iskip
    end %cmd == 'g'


    if cmd == 'k'   % Access to keyboard
            keyboard;

            if sim_flag == 'y'
                    sim_flag = 'd';
            end
```

```
end %cmd == 'k'


if cmd == 'x'   % Accept the data as valid,  Calculate averages and cals
    % ----Sift response to align with avg and combine for net vergence
    v_shift;
    if (input('Accept this data: (y or n) ','s') == 'y')
        n_avg = n_avg + 1;
        data_length = length(avg);
        resp_length = length(test_response);
        if resp_length < data_length
                test_response = [test_response;...
                 test_response(resp_length)*ones(resp_length:data_length,1)];
        end %resp_length < data_length

        for i = 1:data_length
                avg(i) = (avg(i) + test_response(i));
                sum_sq(i) = sum_sq(i) + test_response(i).^2;
        end %i = 1:data_length
        if (n_avg > 2)
                stdm = sum_sq - (avg.^2)/n_avg;
                stdm = sqrt(stdm/n_avg);
        end %(n_avg > 2)

        derivative = zeros(rec_length,1);
        for id = iskip+1:data_length-iskip
                derivative(id,1)  =  (test_response(id+iskip,1)  -  test_response(id-
iskip,1))/scale;
        end %id = iskip+1:data_length-iskip
        acceleration = derv(derivative/n_avg,4);
        for i = 1:data_length
                der_avg(i) = (der_avg(i) + derivative(i));
                sum_sq_der(i) = sum_sq_der(i) + derivative(i).^2;
        end %i = 1:data_length

        if (n_avg > 2)
                std_der = sum_sq_der - (der_avg.^2)/n_avg;
                std_der = sqrt(std_der/n_avg);
        end %(n_avg > 2)

        max_ampl = max(test_response);
        d_sum = (deriv(:,1) + deriv(:,2));
        x_max = max(test_response(1:150));  % Analyze first .75 sec only
        v_max = max(d_sum);
        disp([v_max x_max v_max/x_max])
        if isempty(error)
```

```
                error = [rec_count l_2_cal(1,2) l_2_cal(2,2) r_2_cal(1,2) r_2_cal(2,2)];
                v_x_out = [rec_nu v_max x_max v_max/x_max];
        % ----- Open output file for v_max and x_max ----------------
                out_name = input('Input output file for v_max and x_max: ','s');
                file_out = [export_dir, out_name];
               else
                error   =   [error;   rec_count   l_2_cal(1,2)   l_2_cal(2,2)   r_2_cal(1,2)
r_2_cal(2,2)];
                v_x_out = [v_x_out; rec_nu v_max x_max v_max/x_max];
               end %isempty(error)
          end %(input('Accept this data: (y or n) ','s') == 'y')
           if sim_flag == 'y'
                  sim_flag = 'd';
            end

        end

        if cmd == 'm'  % Multiple  data plot
               dat_plot;
               if sim_flag == 'y'
                      sim_flag = 'd';   % Inhibit simulation this pass only
               end %sim_flag == 'y'
        end %cmd == 'm'


        if cmd == 'd'    % Difference plot.  Required to get isolated data.
               diff_plo;
               if sim_flag == 'y'
                      sim_flag = 'd';   % Inhibit simulation this pass only
               end %sim_flag == 'y'
        end %cmd == 'd'

        if cmd == 't'     %  Insert text into a figure
               fig_nu = input('Figure number: ');
               if fig_nu == 2
                      figure(fig2);
               else
                      figure(fig1);
               end %fig_nu == 2
               tin = input('Input text then click mouse: ','s');
               gtext(tin);
               if sim_flag == 'y'
                      sim_flag = 'd';
               end %sim_flag == 'y'
        end %cmd == 't'
        if cmd == 'e'   % Export data from plots
                fig_nu = input('Output data from Figure number (3 for net verg): ');
                file_name = input('Output file name: ', 's');
                if fig_nu == 1
                        load f1.tmp;
                        eval(['save ', export_dir, file_name,' f1 -ascii']);
                        clear f1;
                elseif fig_nu == 2
                        load f2.tmp;
                        eval(['save ', export_dir, file_name,' f2 -ascii']);
                        clear f2;
                elseif fig_nu == 3
                        verg_sum          =          [t          response(:,1)+response(:,2)
    (deriv(:,1)+deriv(:,2))*.15];
                        eval(['save ', export_dir, file_name,' verg_sum -ascii']);
                        clear verg_sum;
                end %fig_nu == 1
                if sim_flag == 'y'
                        sim_flag = 'd';  % Inhibit this pass only
                end %sim_flag == 'y'
        end %cmd == 'e'

        if cmd == 'p'     % Print one of the two active figures
                fig_nu = input('Input figure number: ')
                if fig_nu == 2
                        figure(fig2);
                else
                        figure(fig1);
                end %fig_nu == 2
                print -dwin;
                if sim_flag == 'y'
                        sim_flag = 'd';   % Inhibit simulation this pass only
                end %sim_flag == 'y'
        end %cmd == 'p'

        if cmd == 'n'   % Enter new  PARAM
                cntl_flag = 'p';
                if sim_flag ~= 'n'
                        sim_flag = 'y';
                end %sim_flag ~= 'n'
                param_nu = 1;
                while param_nu >= 1  & param_nu <= nu_param
                        param_nu = input('Change parameter number: ');
                        if param_nu >= 1 & param_nu <= nu_param
                                x_param = input(['From: ', num2str(x(param_nu)),' to: ']);
                                if isempty(x_param)
```

```
                                        x(param_nu) = 1;
                              else
                                        x(param_nu) = x_param;
                              end %isempty(x_param)
                    end % while param_nu >= 1
          end %param_nu >= 1
end %cmd == 'n'

if sim_flag ~='n' & cmd == 'a' | sim_flag ~= 'n' & cmd == 'q'
          % Save parameters in param
          param(param_pos,:) = [rec_nu, x];  % Save parameters
          eval(['save ', fname_prm, ' param -ascii']);
end %sim_flag ~='n' & cmd == 'a' | sim_flag ~= 'n' & cmd ==

if cmd == 'a'  % Advance record number
          next_rec = 'y';
end %cmd == 'a'

if cmd == 'q'  % Exit
          if exist('file_out') ~= 0

            avg  = [t avg/n_avg der_avg/n_avg stdm std_der acceleration];
            eval(['save ', file_out, '.asc avg -ascii;']);
          end %if exist('file_out') ~= 0
          if exist('file_out1') ~= 0
            eval(['save ', file_out1, '.asc stw_mech -ascii;']);
          end %exist('file_out1') ~= 0
          close; close; fclose all;
          disp('');
          disp('******************** Done ********************');
          return;
end %cmd == 'q'

if cmd == 's' % Save simulations
          output = [t vergence deriv y deriv_sim];
          file_out2 = [export_dir, name];
          eval(['save ', file_out2, '.sim output -ascii;']);


end % cmd == 's'
end %while(1)



end
```

B.2 Combine.m

```
% If you have 2 mat files from the same person, with the same data type already
% categorized into 2 separate files this script puts the matching variables
% together either the vergence ica matrix together, either truncated, or decapped.

clear all;
close all;

file1 = input('Input the filename of the 1st .mat file: ','s');
file2 = input('Input the filename of the 2nd .mat file: ','s');

eval(['load C:\vision_eye_movement_data\ica\temp\', file1]);

if exist('vergence_chop')==1
   tempvc = vergence_chop;
   matchvc = 1;
end

if exist('vergence_trunc')==1;
   tempvt = vergence_trunc;
   matchvt = 1;
end

tempvim = vergence_ica_matrix;


eval(['load C:\vision_eye_movement_data\ica\temp\', file2]);

if exist('matchvc') == 1;
   if exist('vergence_chop')==1
      tempvc2 = vergence_chop;
      vergence_chop = [tempvc,tempvc2];
   else
      disp('Variable Vergence_chop does not match');
   end
end

if exist('matchvt') == 1;
   if exist('vergence_trunc')==1;
      tempvt2 = vergence_trunc;
      vergence_trunc = [tempvt,tempvt2];
   else
      disp('Variables vergence_trunc does not match');
   end
end
```

```
tempvim2 = vergence_ica_matrix;
vergence_ica_matrix = [tempvim,tempvim2];


file3 = input('Save as (filename): ','s');
eval(['save C:\vision_eye_movement_data\ica\', file3]);
```

B.3 Sorta.m

Used to further sort experimental data after categorization done with visresc2call.m

```
%Sorta.m loads file of already categorized vergence data, goes through the
%vergence_ica_matrix or the vergence_trunc or the vergence_chop matrix,
%plotting them, and will let you get rid of the one that you don't want. MLK

clear all;
close all;

nameoffile = input('Input Filename: ','s');
eval(['load c:\vision_eye_movement_data\ica\', nameoffile]);

loop = 'y';

while loop == 'y'
  if exist('vergence_chop') == 1
    data = vergence_chop;
    data_tracker = 'vc';
  elseif exist('vergence_trunc') == 1
    data = vergence_trunc;
    data_tracker = 'vt';
  elseif exist('vergence_trunc')==0
    data = vergence_ica_matrix;
    data_tracker = 'vim';
  end

  [row,col]=size(data);
  figure(1);
  hold;
  clf;
  plot(data);
  disp('What color is the line you want to get rid of?');
  hue = input('b(lue),g(reen),r(ed),c(yan),m(agenta),y(ellow),(blac)k,n(one) ','s');
  if hue == 'b'
    h = 1;
  elseif hue == 'g'
    h = 2;
  elseif hue == 'r'
    h = 3;
  elseif hue == 'c'
    h=4;
  elseif hue == 'm'
    h=5;
  elseif hue == 'y'
  h=6;
elseif hue == 'k'
  h=7;
elseif hue == 'n'
  h = 9999;
end

if h == 9999
  close all;
  loop = 'n';
end

tracker = h;
figure(2);
clf;
if tracker <= col
  figure(2);
  hold on;
  while h <= col;
    p = data(:,h);
    plot(p);
    k = input('Is this the one you want to get rid of?(y/n) ','s');
    if k == 'y'
      tracker = h;
      h = col+1;
    elseif k == 'n'
      h = h+7;
    end
  end
  clf;
  del_tracker = data(:,tracker);
  figure(1);
  hold;
  plot(data);
  plot(del_tracker,'o');
  confirm_del = input('Delete this data?(y/n) ','s');

  if data_tracker == 'vt'
    if confirm_del == 'y'
      data(:,tracker) = [];
    end
    vergence_trunc = data;
  elseif data_tracker == 'vc'
    if confirm_del == 'y'
      data(:,tracker) = [];
    end
```

```
            vergence_chop = data;
         elseif  data_tracker == 'vim'
            if confirm_del == 'y'
               data(:,tracker) = [];
            end
            vergence_ica_matrix = data;
         end

         figure(1);
         hold;
         clf;

         if data_tracker == 'vt'
            plot(vergence_trunc);
         elseif data_tracker == 'vc';
            plot(vergence_chop);
         elseif data_tracker == 'vim'
            plot(vergence_ica_matrix);
         end

         eval(['save c:\vision_eye_movement_data\ica\', nameoffile]);
         loop = input('Did you want to delete another from this file?(y/n) ','s');
      end
   end
```

B.4 Sortb.m

```
%Sortb.m loads file of already categorized vergence data, goes through the
%vergence_ica_matrix plotting them, and will let you get rid of the one
%that you don't want. MLK

clear all;
close all;

nameoffile = input('Input Filename: ','s');
eval(['load c:\vision_eye_movement_data\ica\', nameoffile]);

loop = 'y';


while loop == 'y'
%   if exist('vergence_chop') == 1
 %      data = vergence_chop;
  %     data_tracker = 'vc';
  % elseif exist('vergence_trunc') == 1
  %     data = vergence_trunc;
   %    data_tracker = 'vt';
   %  elseif exist('vergence_trunc')==0
        data = vergence_ica_matrix;
        data_tracker = 'vim';
        %end

   [row,col]=size(data);
   figure(1);
   hold;
   clf;
   plot(data);
   disp('What color is the line you want to get rid of?');
   hue = input('b(lue),g(reen),r(ed),c(yan),m(agenta),y(ellow),(blac)k,n(one) ','s');
   if hue == 'b'
      h = 1;
   elseif hue == 'g'
      h = 2;
   elseif hue == 'r'
      h = 3;
   elseif hue == 'c'
      h=4;
   elseif hue == 'm'
      h=5;
   elseif hue == 'y'
      h=6;
```

```
    elseif hue == 'k'
       h=7;
    elseif hue == 'n'
       h = 9999;
    end

    if h == 9999
       close all;
       loop = 'n';
    end

    tracker = h;
    figure(2);
    clf;
    if tracker <= col
       figure(2);
       hold on;
       while h <= col;
          p = data(:,h);
          plot(p);
          k = input('Is this the one you want to get rid of?(y/n) ','s');
          if k == 'y'
             tracker = h;
             h = col+1;
          elseif k == 'n'
             h = h+7;
          end
       end
       clf;
       del_tracker = data(:,tracker);
       figure(1);
       hold;
       plot(data);
       plot(del_tracker,'o');
       confirm_del = input('Delete this data?(y/n) ','s');

%      if data_tracker == 'vt'
%         if confirm_del == 'y'
%            data(:,tracker) = [];
%         end
%         vergence_trunc = data;
%      elseif data_tracker == 'vc'
%         if confirm_del == 'y'
 %           data(:,tracker) = [];
 %        end
%         vergence_chop = data;
%       elseif  data_tracker == 'vim'
          if confirm_del == 'y'
             data(:,tracker) = [];
          end
          vergence_ica_matrix = data;
          %      end

       figure(1);
       hold;
       clf;

       %if data_tracker == 'vt'
       %   plot(vergence_trunc);
       %       elseif data_tracker == 'vc';
       %   plot(vergence_chop);
       %       elseif data_tracker == 'vim'
          plot(vergence_ica_matrix);
          %end

       eval(['save c:\vision_eye_movement_data\ica\', nameoffile]);
       loop = input('Did you want to delete another from this file?(y/n) ','s');
    end
end
```

B.5 Truncate.m

```
%truncate.m is a script that reads in the vergence_ica_matrix, and then truncates it at
% 401 to prepare it for ICA analysis


con = 'y';
while  con == 'y';
   clear all;
   close all;
   filenamet = input('Input the filename of the .mat file: ','s');
   eval(['load C:\vision_eye_movement_data\ica\', filenamet]);
   a = vergence_ica_matrix;
   [r,c] = size(a);
%n data point that user wants to truncate at, for row truncation
%n = input('Truncate at data point: ')
   n = 401; %Automated Truncation to 2 seconds instead of asking for user input
   a(n:r,:)=[];
   vergence_trunc = a;
   eval(['save C:\vision_eye_movement_data\ica\', filenamet]);

%clear all;
%filename = input('File name: ','s');

   eval(['load C:\vision_eye_movement_data\ica\', filenamet]);
%whos
   figure;
   hold;
   plot(vergence_trunc);
   con = input('Would you like to truncate another data file? (y/n) ','s');
end
```

B.6 Edited excerpt from icamt1.m

```
%Michele edited to save results in a file with with partition label here:
 lding_save = datasource;
 partition_save = int2str(partition);
 lding_save = [lding_save, 'c', partition_save];
 %eval(['save c:\vision_eye_movement_data\ica\semmlow\2comps\', ica_save]);
 eval(['save C:\vision_eye_movement_data\ica\semmlow\2comps\', lding_save,' lding']);
 %End Michele Edit
```

B.7 F_loaderm.m

MATLAB script written to use with FastICA package to easily load variables into FastICA

```
%F_LoaderM will ask the name of the file, and load the variable that it
%is told to load.

clear all;
data_source_type = input('Is Data Simulated? (y/n) ','s');
filename = input('File Name: ','s');
if data_source_type == 'y'
   eval(['load c:\vision_eye_movement_data\ica\simulated\', filename]);
   data = data';
   %data(:,1)=[];
   %eval(['save c:\vision_eye_movement_data\ica\simulated\', filename]);
else
   eval(['load c:\vision_eye_movement_data\ica\', filename]);
end
whos;
```

B.8 F_saverm.m

MATLAB script written to use with FastICA package to easily save variables created by FastICA

```
%f_saverm asks for the filename, how many components the user ran to,
%and the name of the filename the user wants everything saved to.
%then it saves it to the directory with the new file name with

comps = input('How Many components was ICA run to? ');
ica_save = input('Input file to save to: ' , 's');

if comps == 2;
   eval(['save c:\vision_eye_movement_data\ica\2comps\', ica_save]);
elseif comps ==3;
   eval(['save c:\vision_eye_movement_data\ica\3comps\', ica_save]);
end

%comps = input('How Many components was ICA run to? ');
%ica_load = input('Input file to that you save to: ' , 's');

%if comps == 2;
%   eval(['load c:\vision_eye_movement_data\ica\2comps\', ica_load]);
   %elseif comps ==3;
   %eval(['load c:\vision_eye_movement_data\ica\3comps\', ica_load]);
   %end

%whos
```

B.9 Icaf_loaderm.m

MATLAB script written to use with FastICA package to easily load variables created by FastICA

```
%F_LoaderM will ask the name of the file, and load the variable that it
%is told to load.  Then the user will be prompted to call the fasticag
%function.

clear all;
close all;

ica_filename = input('Input name of file to be loaded: ', 's');
nu_comps = input('How Many components was ICA run to? ');


if nu_comps == 2;
   eval(['load c:\vision_eye_movement_data\ica\2comps\', ica_filename]);
elseif nu_comps ==3;
   eval(['load c:\vision_eye_movement_data\ica\3comps\', ica_filename]);
end

whos;
```

REFERENCES

1. Montgomery, T.M., http://www.tedmontgomery.com/the_eye/. p. Anatomy, Physiology & Pathology of the Human eye.

2. Semmlow, J.L. and W. Yuan, *Adaptive modification of disparity vergence components: an independent component analysis study.* Invest Ophthalmol Vis Sci, 2002. **43**(7): p. 2189-95.

3. Semmlow, J.L., G.K. Hung, and K.J. Ciuffreda, *Quantitative assessment of disparity vergence components.* Invest Ophthalmol Vis Sci, 1986. **27**(4): p. 558-64.

4. *Eye anatomy - muscles,* http://www.wa-eyemd.org/anatomy-muscles.htm.

5. Mays, L.E., et al., *Neural control of vergence eye movements: neurons encoding vergence velocity.* J Neurophysiol, 1986. **56**(4): p. 1007-21.

6. Rashbass, C. and G. Westheimer, *Disjunctive Eye Movements.* J. Physiol, 1961. **159**: p. 339-360.

7. Semmlow, J.L. and W. Yuan, *Components of disparity vergence eye movements: application of independent component analysis.* IEEE Trans Biomed Eng, 2002. **49**(8): p. 805-11.

8. Hung, G.K., J.L. Semmlow, and K.J. Ciuffreda, *A dual-mode dynamic model of the vergence eye movement system.* IEEE Trans Biomed Eng, 1986. **33**(11): p. 1021-8.

9. Mays, L.E., *Neural control of vergence eye movements: convergence and divergence neurons in midbrain.* J Neurophysiol, 1984. **51**(5): p. 1091-1108.

10. Semmlow, J. and N. Venkiteswaran, *Dynamic accommodative vergence components in binocular vision.* Vision Res, 1976. **16**(4): p. 403-10.

11. Hung, G.K., J.L. Semmlow, and K.J. Ciuffreda, *Identification of accommodative vergence contribution to the near response using response variance.* Invest Ophthalmol Vis Sci, 1983. **24**(6): p. 772-7.

12. Hyvarinen, A., *Fast and Robust Fixed-Point Algorithms for Independent Component Analysis.* IEEE Trans on Neural Networks, 1999. **10**(3): p. 626-634.

13. Hyvarinen, A., *Survey on Independent Component Analysis*, in *Neural Computing Surveys*. 1999. p. 94-128.

14. *ICA demo, http://www.cis.hut.fi/projects/ica/icademo/,* ICA Group at Helsinki University of Technology.

15. *FastICA package for Matlab, http://www.cis.hut.fi/projects/ica/fastica/,* ICA Group of Helsinki University of Technology.

16. Semmlow, J.e.a., *multi_sim_sym.m*. p. Creates Simulated vergence data.

17. Semmlow, J., *icamt1.m*. p. ica program uses fastica, uses partitioning, and scaling to 2 components only.

18. Cambridge Research Systems, I., *Skalar IRIS IR Light Eye Tracker http://www.crsltd.com/catalog/skalar/iris/*. 1999, Cambridge Research Systems, Inc.

19. Alvarez, T.L., Chua, F. B., Daftari, A. P., DeMarco, R. M., Herrera, L., Bergen, M. T., *Vision Research Program*. 2003. p. Labview VI.

20. Alvarez, T.L., *Vision Data Analysis Program*. 1998,2002.

21. Kung, M., T. Alvarez, and J. Semmlow. *Interaction of Disparity and Accommodative Vergence*. in *IEEE 29th Annual Northeast Bioengineering Conference*. 2003. Newark, NJ.

22. Hung, G.K. and J.L. Semmlow, *A quantitative theory of control sharing between accommodative and vergence controllers*. IEEE Trans Biomed Eng, 1982. **29**(5): p. 364-70.