Application of MEMS Inertial Sensors in Sensing Passive Eye Response as a Surrogate for Brain Response to Head Acceleration and Rotation for On-field Objective Assessment of Concussion

by

Yuan Meng

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 15, 2018

Keywords: mTBI, concussion, passive eye response, on-field diagnosis, MEMS IMU, data fusion

Approved by

Mark L. Adams, Chair, Assistant Professor of Electrical and Computer Engineering
Robert N. Dean, McWane Professor of Electrical and Computer Engineering
Thaddeus Roppel, Associate Professor of Electrical and Computer Engineering
Stuart Wentworth, Associate Professor of Electrical and Computer Engineering

Abstract

The passive eye response (PER) may act as a surrogate for the brain response to head acceleration during a concussion causing impact. A novel eye movement sensor designed in this dissertation is used to verify this hypothesis and to predict the risk of mild traumatic brain injury (mTBI). Microelectromechanical systems (MEMS) accelerometers are proven to capture the relative displacement of the eye to the bony socket/skull in a similar fashion to the brain. Expansion to dynamic systems is achieved by applying 6-DoF MEMS inertial measurement units (IMU), which are tested on a skull-brain-eye model and human volunteers in drop-and-impact experiments. An advanced sensor fusion technique is designed and applied in processing the IMU data. Similar angular accelerations of eye and brain relative to skull are observed in the IMU data during rotation tests. Strong correlations of eye and brain accelerations are discovered in the drop-and-impact model tests suggesting that sensing the PER using IMU's could provide better outcomes than sensing head acceleration for real-time on-field objective mTBI monitoring, assessment, and diagnosis.

Acknowledgments

I would like to take this opportunity to give thanks to Dr. Mark L. Adams for his enthusiastic help, wise guidance, and selfless support during my pursuing Ph.D. He is not only a smart, knowledgeable and responsible advisor, but also an honorable career and life mentor for me. I would like to thank Dr. Robert Dean for his introduction to this research project, without which none of the current achievements could be made. Also thanks to him for his loving help and precious suggestions on paper composition. Thank Dr. Lei Liu and Dr. Mark Bolding for their sponsorship for the project, professional consulting, help in paper modification and grant proposal writing, and their brave hearts to volunteer in experiments. Thank Dr. Stuard Wentworth for his TA sponsorship and inspiring instruction in teaching.

I would also like to thank all my colleagues at STORM Lab, especially Brent Bottenfield, J. Craig Prather, Michael Bolt, and Tyler Horton for our happy cooperation and great teamwork in the project.

Special thanks to Linyuan Zhang, my brilliant, beautiful and amazing wife, for her constant encouragement, enduring love and unwavering support, and to my parents Yanjun Meng and Zhihua Chen, my grandparents Shouyun Chen and Guizhi Liu, for all their selfless love and constant support and sacrifice both financially and emotionally. Also thanks to all my brothers and sisters in Christ for their love, support and prayers. Thank God for all.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Auburn	Auburn University
BPF	band pass filter
CT	computed tomography
CTE	chronic traumatic encephalopathy
DAQ	data acquisition
DoF	degree of freedom
FB	front brain
FFT	fast Fourier transform
FSK	frequency-shift keying
HACC	head acceleration
HIT	Head Impact Telemetry
IMU	inertial measurement unit
LE	left eye
LFCSP	leadframe-based chip-scale package
LGA	land grid array
LSB	least significant bit
MB	middle brain

MCU microcontroller unit

MEMS microelectromechanical systems

MRI magnetic resonance imaging

mTBI mild traumatic brain injure

ODR output data rate

PB posterior brain

PCB printed circuit board

PDMS polydimethylsiloxane

PER passive eye response

PVC polyvinyl chloride

RAM random-access memory

RE right eye

RF radio frequency

SRS Sideline Response System

STD standard deviation

STORM Lab Sensors, Transducers, Optics, RF and MEMS Lab

TBI traumatic brain injury

UAB the University of Alabama at Birmingham

VOR vestibulo-ocular response

Chapter 1

Introduction

In recent years, public awareness of the risks of concussive trauma in contact-sport-related activities has increased dramatically. Since 2009, all 50 states and the District of Columbia have enacted legislation regulating concussion treatment [1]. The prevalence of concussions in particular sports has increased, though it is not known whether this is the result of increased incidence of injury or improved diagnostics [2]. Previous research indicates the prevalence of concussions could be between 1.6 and 3.8 million injuries per year [3].

A concussion is often referred to by doctors as a mild traumatic brain injure (mTBI) , which is caused by the displacement and deformation of the brain in the skull during head or body impacts [4]. Both terms are used when a person experiences a change in normal brain function for no longer than a minute following trauma. Concussion/mTBI is such a high prevalence injury that athletes in contact sports at all levels are at high-risk. Though usually not being life-threatening, its effects can be serious and life-long. The immediate affects of mTBI may include one or more of the following symptoms: headache, nausea and vomiting, loss of consciousness, change in vision, being confused, hardness to arouse, memory loss (amnesia), and feeling of “lost time”, etc.

Because mTBI is mostly diagnosed subjectively and the symptoms can often mirror symptoms of other pathologies, objective diagnosis can be challenging but is critical as undiagnosed thus untreated mTBI can result in a number of neurological and cognitive conditions such as issues with memory and reasoning, difficulties with balance and sight, weakened communication skills, and emotional instability. Repeated mTBI can result in the accumulation of these symptoms over time. Additionally, mTBI can cause epilepsy and increase the risk for age-related neurological disorders like Parkinson's and Alzheimer's disease [5]. Details on concussion/mTBI are given in Section 2.1.

The ability for real-time assessment of brain motion and deformation during head impact is of immense value in understanding the biomechanics of mTBI, in assessing mTBI risk, and in diagnosing mTBI objectively. The ability to objectively assess the risk of mTBI in the field of play or on the sideline is also very desirable for making important decisions such as return-to-play. However, there are major challenges to real-time diagnosis. First, while advanced technologies such as magnetic resonance imaging (MRI) and computed tomography (CT) have the potential to image brain motion/deformation during small, non-injurious head accelerations [6–8], they cannot be used to evaluate the large, potentially injurious head acceleration encountered in the field. Second, simultaneous recording of brain motion/deformation has been a challenge because the fluidic human brain moves quite differently than the solid skull moves during an impact (Figure 1.1 [9]). Insertion of sensors inside the skull is not applicable on a healthy person however vulnerable to concussion he or she is. This issue is further discussed in Section 2.2.

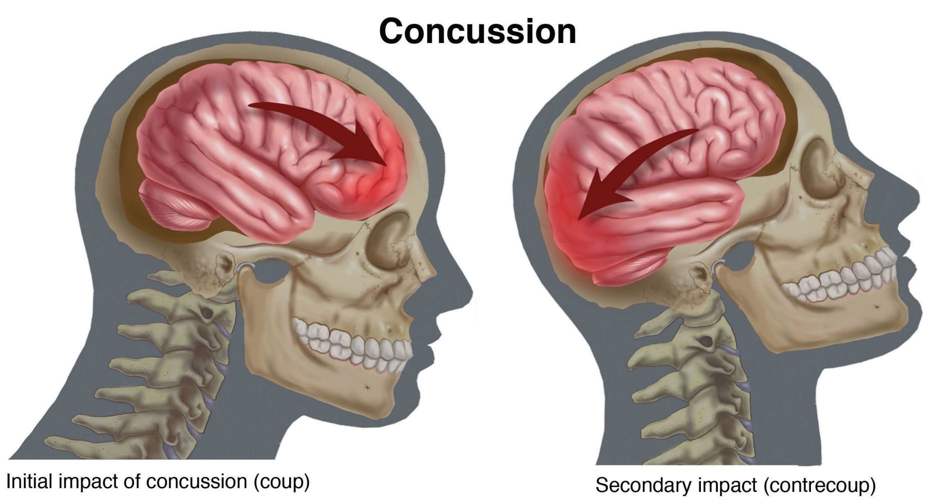


Figure 1.1: An Illustration of Human Brain Concussion.

A significant effort toward this direction is the Head Impact Telemetry (HIT) System^{TM}, which uses an array of non-orthogonal accelerometers mounted inside helmets to provide in-field real-time monitoring of head accelerations with the intention to predict mTBI from head kinematics [10–13]. While the HIT system has made important contributions to our understanding about head impacts in sports, its ability to predict mTBI has not been established [14–20]. The likely reason is that head accelerations may not directly translate to brain motion/deformation in the skull. More discussions on HIT system are given in Section 2.3.

An alternative approach to characterizing brain response to impact is through the passive response of the eye. After impact, extraocular muscles controlling eye movement are unable to react for approximately 20 ms [21]. During this period it is hypothesized that the eye moves analogously to the brain as tissue suspended in fluid, but different relative to the movement of the skull. Objective in-field real-time monitoring of the motion and the deformation of the brain therefore can be realized by indirectly sensing the motion/deformation of the eye. The goal of this research is to test the feasibility of predicting the risk of mTBI using the passive mechanical response of the eye as a surrogate for the brain's response to a head blow.

The design of experiment in Section 4.1 has been done at Auburn University and confirmed the use of microelectromechanical systems (MEMS) accelerometers in measuring the relative displacements of polydimethylsiloxane (PDMS) model eyes relative to their 3D-printed model sockets when the sockets are hit by a suspended polyvinyl chloride (PVC) hammer. Acceleration data were integrated to yield total displacement of both eye and socket model. Though MEMS accelerometers are able to measure the relative displacement in the stable socket model systems, more attitude information is needed for accurate estimation in dynamic systems where the total displacements are no longer negligible. The work, as introduced in Section 4.2, expanded upon these findings by utilizing MEMS inertial measurement units (IMU) to obtain angular, acceleration, velocity, and displacement data in a simulated impact test for eye response versus head response. The simulation utilized a 3D-printed skull model filled with a model gelatin brain. Additionally, the model includes scale eye replicas composed of PDMS and gelatin eye sockets. The skull model was subjected to a number of drop tests in which IMU's were placed on the skull as a reference and on the eyeball under an artificial eyelid. In addition, the investigators volunteered to conduct human tests to verify the eye-skull model and sensor response. The same make of IMU's and design of eye-skull model are used in head-rotation experiments in Chapter 5, extending the use of IMU's in sensing passive eye rotation as a surrogate for brain rotation. Tests are done on an air/vacuum-actuated head turner and the vestibulo-ocular response (VOR) chair at the University of Alabama at Birmingham (UAB) . After validating the usage of MEMS IMU's in sensing relative linear and rotational accelerations to the head, correlations between the passive eye and brain motions are studied in Chapter 6 by model experiments with IMU's embedded in the gelatin phantom brain. Finally, summary and conclusion are given in Chapter 7, and future work in Chapter 8.

Chapter 2

Literature Review

2.1 Traumatic Brain Injury (TBI) and mild TBI

Traumatic Brain Injury (TBI) is a high prevalence injury [22,23], especially in high-risk populations such as athletes involved in contact or high-speed sports. There are roughly three types of TBI according to the impact sources: direct impact injury, acceleration/deceleration injury, and shock wave injury (Figure 2.1 [24]). Up to 3.8-million sport-related TBIs occur in the US each year [25]. Among collegiate and high-school sports, American football has the largest number of TBI cases, due to the large number of participants and the high frequency of body, especially head, impacts [25, 26]. According to Meehan III et al. [27], nearly one-third of TBIs go undiagnosed because of the lack of the highly noticeable signs

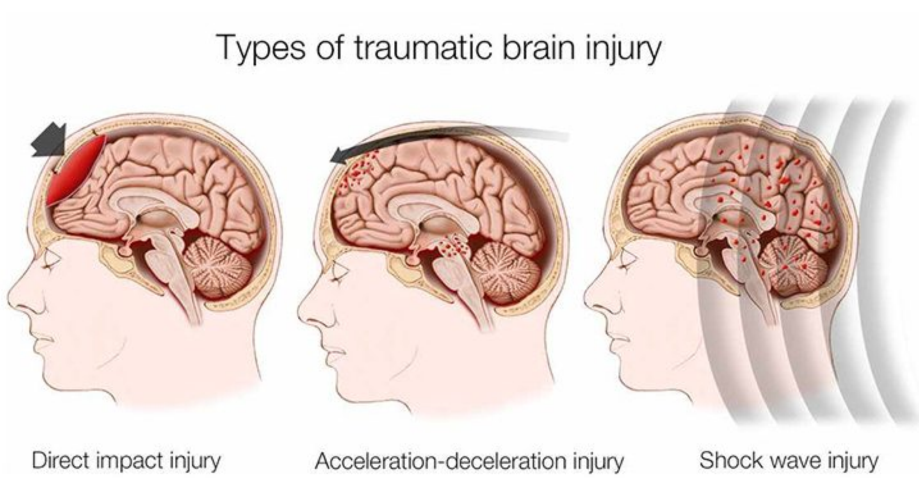


Figure 2.1: Types of Traumatic Brain Injury (TBI).

and symptoms. Many times athletes may also minimize the severity of their injuries to remain in the competition.

Although 90% of TBIs are mild in severity (mTBI or concussion) [26] and the patients usually recover quickly, there are concerns about the long-term effects of repetitive sub-concussive blows, especially when they are not diagnosed and treated in time. Undiagnosed mTBIs increase lost time from school [28] and risk of death because of the Second Impact Syndrome [29]. Their cumulative effects seem not only to slow down recovery, but also to dramatically increase the possibility of long-term neurophysiological impairment and depression. They carry risks for cognitive decline [30], depression [31], suicide [32], dementia [33], Parkinson's disease [34], and Alzheimer's disease [35] etc. Professional contact/high-speed-sport athletes also have a large number of cases where terminal neurological degenerative diseases like chronic traumatic encephalopathy (CTE) and young-age amyotrophic lateral sclerosis were caused by the mTBI [36, 37]. Last but not least, crucial administrative decisions during the games, such as return to play, have to be made in real time on the field to reduce the risk of repeated injury.

The pathogenesis of mTBI is the mechanical strain in the brain tissue associated with the transient motion and deformation of the brain inside the skull induced by injurious impacts. It is the consensus that most sport-related mTBIs are the results of impulsive forces from head, neck or body blows that cause sudden acceleration/deceleration of the brain [38]. Animal studies demonstrated that both linear and rotational head accelerations (HACC) can cause mTBIs, but with different manifestations [39, 40]. Physical simulations using gelatin-filled brain models demonstrated that the largest shear strains in the brain models were associated with rotations of the head [41–43]. Empirical studies of brain displacement and deformation in animal models and human cadavers were conducted by direct observation of the brain through transparent polymer calvariums, by implanting accelerometers in the brain or by high-speed x-ray imaging of lead markers inserted in the brains of cadavers [44–51]. These studies demonstrated that the brain, which is fluid rather than solid, tends to keep its position and shape when the head begins to accelerate, and to move and deform as the head motion slows, reaches steady state or changes direction [45, 46]. Brain tissues at different locations in the skull were displaced by different amounts, with local peaks of deformation and shear strain in the experiments. In human cadavers, HACC and speed peaked at ∼5 ms after impact and maximum strain peaked at ∼10 ms [46]. It was postulated that the brain "is injured when its constituent particles are pulled so far apart that they do not join up again properly when the blow is over" [41]. Brain response in living human participants applied with impacts can also be studied using tagged MRI [6–8]. Despite the fact that tagged MRI is not suitable for real-time on-field monitoring of mTBIs, this technology can be used to validate other techniques buy comparing results from duplicated experiments. Low linear or angular HACC (1-3 g or 100-300 rad/s^2) were shown to produce linear or circumferential brain displacements (2-3 mm or 1-2^o) and shear strains (0.02-0.06) [52–55]. Finite element model simulations have also been used to study the relationship between HACC and the strain of brain tissue [56–61]. The evidence suggests that the strain within brain tissue associated with brain acceleration/deceleration is the cause of mTBI.

2.2 Objective, On-Field Monitoring of Head Acceleration

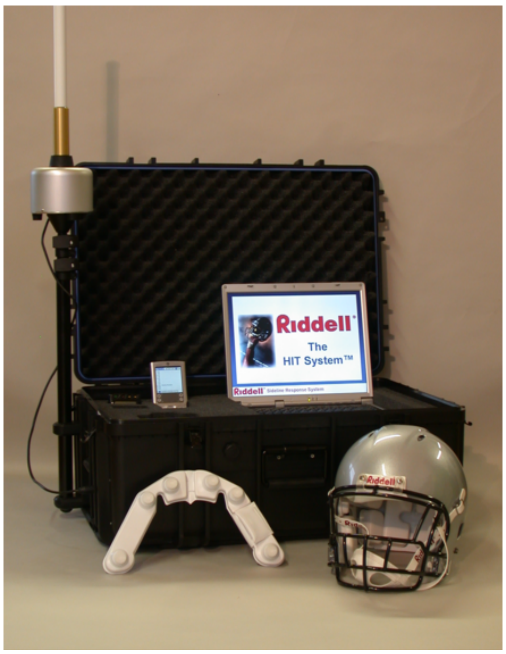
The brain motion/deformation data cited above was acquired under conditions where the magnitude, direction, location and dynamics of head impacts were strictly controlled. Brain injuries in real life, however, are affected by many different factors, which laboratory research may not be able to simulate. Factors such as readiness for impact, protection gear, head impact v.s. body blow (whiplash), etc. are known to influence the risk of mTBI. To make diagnosis more difficult, mTBI lacks the overt signs of TBI, such as a fractured skull and/or prolonged loss of consciousness. It is defined primarily by symptoms reported by the patients themselves who have received the concussive blow [38]. However, voluntary reporting is not reliable. For instance, according to a survey of a total of 1532 varsity football players from 20 high schools in the Milwaukee, Wisconsin as shown in Table 2.1 [62], more

Concussion Reporting Data*		Percentage of Subjects
Concussion Reported to	Certified athletic trainer	76.7
	Coach	38.8
	Parent	35.9
	Teammate	27.2
	Other (eg, family physician, student)	11.7
Why Concussion not Reported	Did not think it was serious enough	66.4
	Did not want to leave the game	41.0
	Did not know it was a concussion	36.1
	Did not want to let down teammates	22.1
	Other reasons	9.8
*Categories are not mutually exclusive; subjects were asked to check all that apply.		

Table 2.1: The Frequency Distribution for Concussion Self-reporting and Reasons Why Concussions Not Reported.

than 50% of high-school football players who sustained concussions did not report their mTBI-related symptoms, because they did not consider the injury to be serious enough or did not want to be withheld from competition. Therefore, objective assessment of mTBI risk is vitally important for identifying these players. Such assessment is most effective if taken on-field, right after impact, real-time if possible, so that potentially concussive blows can be detected, and thus repeated injuries can be prevented. The effectiveness of protective equipment can be realistically evaluated and techniques to prevent injurious contacts can be taught. Because direct measurement of brain dynamics during impact is extremely difficult in humans and it is impossible to assess brain responses to concussive blows in a live human being outside of laboratory settings, the current focus of on-field mTBI research naturally falls on the head kinematics, i.e., the linear and angular accelerations of the head, simply because it can be more readily assessed [63]. This has been the motivation of a large body of studies concerning the relationship between head/body blows, HACC and the risk of mTBI.

Instrumented helmets and bite-plates/mouthguards have been used to measure HACCs in human volunteers during various sports and daily activities [64–70]. The state of the art in on-field head kinematics assessment is the Head Impact Telemetry (HIT) System by Simbex^{TM}, which is now marked by Riddell^{TM} in the Riddells Sideline Response System (SRS) (Figure 2.2a) and the InSite Impact Response System [71, 72] (Figure 2.2b). The HIT system features 6 spring-loaded accelerometers mounted inside each helmet as shown in Figure 2.2c, real-time recording of head kinematics (such as location of head impact, peak



(a)

(b)

(c)

Figure 2.2: The Head Impact Telemetry (HIT) System: (a) HIT System Apparatus; (b) HIT System in Riddell InSite; (c) The Six Accelerometers of HIT System Embedded in a Helmet.

HACCs and impact duration), wireless transmission of time-stamped data to a collector on the side line that converts the data to head impact severity measures and an acoustic alert for potentially injurious impacts. The HIT System has been validated against the Hybrid III dummy [11], has been used in a wide variety of sports with more than 2,000,000 sets of impact data collected from high-school and college athletes [71] but has not been used in professional games. The HIT achievements include quantitative assessments of position-specific and session-specific (game and practice) impact frequencies, mean peak linear and angular accelerations, the spectra of linear and angular accelerations and assessments of concussion thresholds for collegiate and high-school football players [73–75]. These studies documented how the head moved under impact, but not how the brain responded.

Another typical position to instrument on-field concussion assessment systems is in a mouthguard. Other groups are working to design more accurate systems such as the integration of sensors on a mouthguard to monitor dangerous impacts [76–78]. Figure 2.3 shows such a device produced by Prevent Biometrics™ [79]. Mouthguard-embedded systems expand the applications of on-field concussion assessment to more helmetless sports such as

Figure 2.3: A Mouthguard Instrumented Concussion Sensing Device by Prevent Biometrics.

rugby and boxing etc., however, the measurands are still the same as the helmet-mounted ones: head kinematics. While these studies and products have improved our understanding of how, where and when athletes get hit during games, it has become clear that head kinematics alone cannot predict the risk of mTBI. What is missing is the capability for on-field real-time assessment of how head impact load is translated into brain displacement and deformation in the skull. One way to bridge the gap is by monitoring surrogates that undergo similar motion and deformation as the brain during a head/body blow.

2.3 Eye Tracking Technologies

Many researchers have realized the potential to use eye movements to diagnose concussion. They use conventional, video-based eye trackers to compare eye movement patterns and dynamics of normal controls to concussion suspects or patients [80–84]. However, these studies differ from the proposed eye sensor in several ways. Firstly, as shown in Figure 2.4, eye trackers such as the state-of-art EyeLink^{TM} 1000 Plus [85] require stable head position to achieve a high measurement accuracy. Even for the wearable version of EyeLink^{TM} II (Figure 2.5 [86, 87]), special devices, such as the head- and chin-rest and the head camera are introduced to minimize head motion and to compensate the effects of head motion. These devices are useless under the head acceleration condition studied in this research. The velocity and amplitude are too much for these eye trackers to operate normally. Secondly, current concussion eye movement studies differ from this research in their purposes. They study eye movement abnormalities after concussive head impacts. The purpose for developing the eye sensor is to understand how the brain is injured in the skull and to assess concussion risk during head impact without opening the skull. The eye is a surrogate of the brain. Thirdly, current concussion eye movement studies measure different kinds of eye movements from the ones interested in this research. Instead of the eye rotations under neural control, the eye motion studied here occurs only during a short time window during a head impact during which neural control has not activated, which is further discussed in the next chapter. This passive eye motion may mirror the brain motion/deformation during the same impact. An injury threshold may thus be derived to predict brain injury. Fourthly, current concussion eye movement studies have different applications. They are used in clinical settings long after head impacts to determine if there is a concussion. The ultimate goal of the eye sensor is real-time, remote assessment of concussion risk in the field. It can assist the prediction and diagnosis of concussion and it can also provide objective evidence to assist time-critical decisions such as return-to-play or med-evac in the field. Lastly, the final product of the eye sensor will be untethered and will be comfortable to wear.

Figure 2.4: The Setup of EyeLink 1000 Plus.

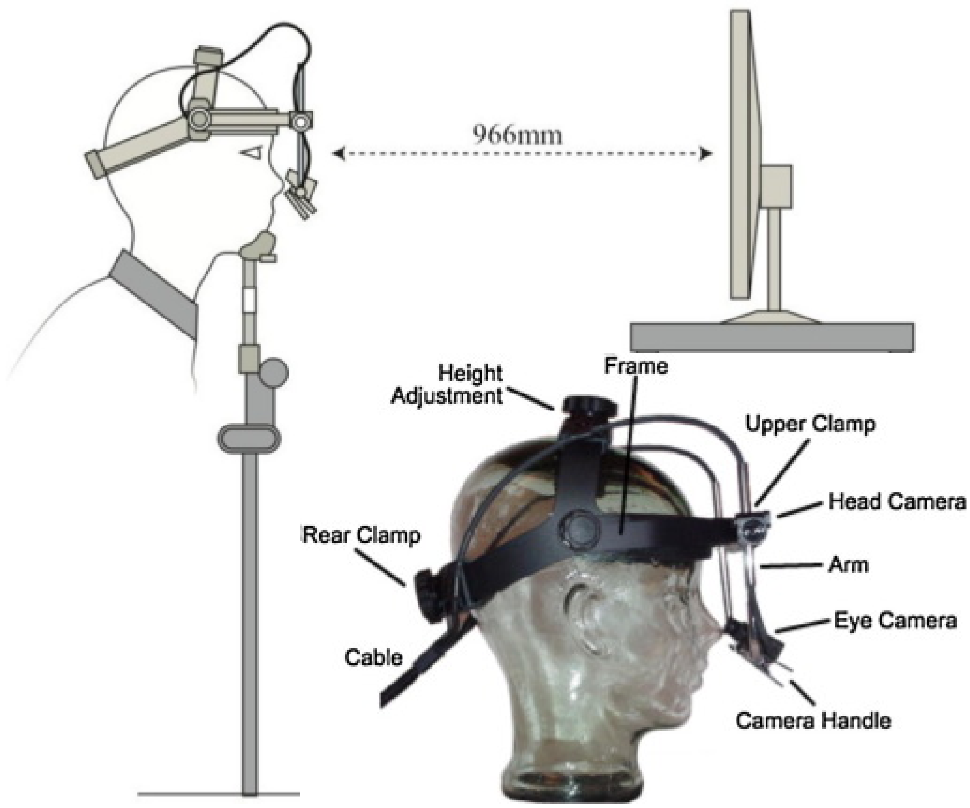


Figure 2.5: The Setup of EyeLink II.

Chapter 3

Theory Basis and Discussion

3.1 Head Kinematics Cannot Predict Concussion

Although the HIT System is not designed to be a diagnostic tool for mTBI [88], one of its designated purposes is to detect potentially injurious head impacts [18, 71]. However, head-kinematics-based severity criteria for mTBI risk have been found to be incapable of separating concussive from non-concussive players. For any such criteria, there were always many more severe HIT impacts that did not produce mTBI and there were mTBI cases that were caused by much less severe HIT impacts [18]. In a 3-year study of 88 collegiate football players wearing HIT embedded helmets, no significant relationships were found between HIT data and clinical assessments collected within 24-48 hours post-injury [14]. Among the more than 100,000 HIT head impact data sets collected from 95 high-school football players during 4 years, no significant correlations were found between HIT impact characteristics and athlete demographics (previous number of concussions), symptoms, ImPACT (a concussion management software [89]) pre/post injury change scores or the number of days until recovery [16]. No significant differences in peak linear HACC levels and in frequencies of impacts were found among college football players with and without clinically-observed impairment and "functionally-observed impairment" [17]. High-school football players who had clinically diagnosed concussions accumulated significantly fewer head collisions reported by the HIT system than their teammates who had no concussion in all collision intensity categories and locations [20]. Researchers have realized that "the efficacy of using helmet telemetry to identify concussion and/or concussion-like signs and symptoms in the absence of subjective information provided by the athlete was not demonstrated" [15], that "the HIT data on the injury are not accurate" [12] and that "impact severity (measured in acceleration/deceleration) may be clinically irrelevant" [63]. Adding more sensors to the helmet may not result in fundamentally different assessments [90, 91]. It has become clear that HACC is only a risk factor of mTBI [63]. It alone is not sufficient in predicting mTBI [14, 19].

3.2 Using the Eye as a Surrogate to Infer Brain Response to Impact

mTBI is the consequence of a series of events: head/body impact → linear/angular head acceleration → brain acceleration and deformation → brain tissue shear → mTBI [92]. The fact that HACCs recorded on-field by the helmet or mouthguard mounted sensors have failed to predict mTBI occurrence shows an urgent need to explore new ways to assess on-field real-time transmission of the head/body impact loading to the brain displacement/deformation and tissue shear. As mentioned above, none of the currently available methods is capable of doing so.

This research uses the eye as a surrogate to infer the brain response to impacts. As an extension of the brain, the eye sits inside a bony socket, tethered by the optical nerve and the six extraocular muscles (Figure 3.1 [93]) and cushioned by several types of orbital tissues. It is subject to the same forces as the brain during head/body impacts. While the normal voluntary and reflexive movements of the eye are neurally-controlled rotations about its center of rotation, there is solid evidence that during a short time window immediately after a HACC, the eye undergoes a passive motion before neural control takes over to generate a vestibulo-ocular response (VOR). During this time window, the eye's motion is governed purely by laws of dynamics [21, 94]. This passive eye response (PER) can be seen as the entire ocular mass, eyeball, extraocular muscles, nerves, vessels and orbital tissues, interacting with the eye socket as a whole under the forces transmitted from HACC. Because the eye is much more accessible than the brain which is sealed in the skull, physical measures of PER to head/body impacts can potentially be used to infer the brain response to impacts.

If such a link can be established, a window is opened for objective, on-field, real-time assessment of brain responses in sport fields or even war zones. New insights can be brought to the relationship between real-life impacts, both benign and injurious, and brain responses. Furthermore, because MEMS sensors embedded in small ocular inserts can be placed inside the pouches around the eye, a completely new device for real-time objective monitoring of brain response to impact in the field can be developed for the purposes of mTBI detection, assessment, management and prevention.

3.3 Passive Eye Response Immediately after Head Acceleration

Vestibulo-ocular responses (VOR) are rapid reflexive rotations of the eyes for the purpose of stabilizing eye gaze during head/body motions [95]. The direction of VOR is therefore compensatory, i.e. in the opposite direction of head/body motion. VORs are driven by crossed excitation and inhibition of the signals from the semi-circular canals and the otolith

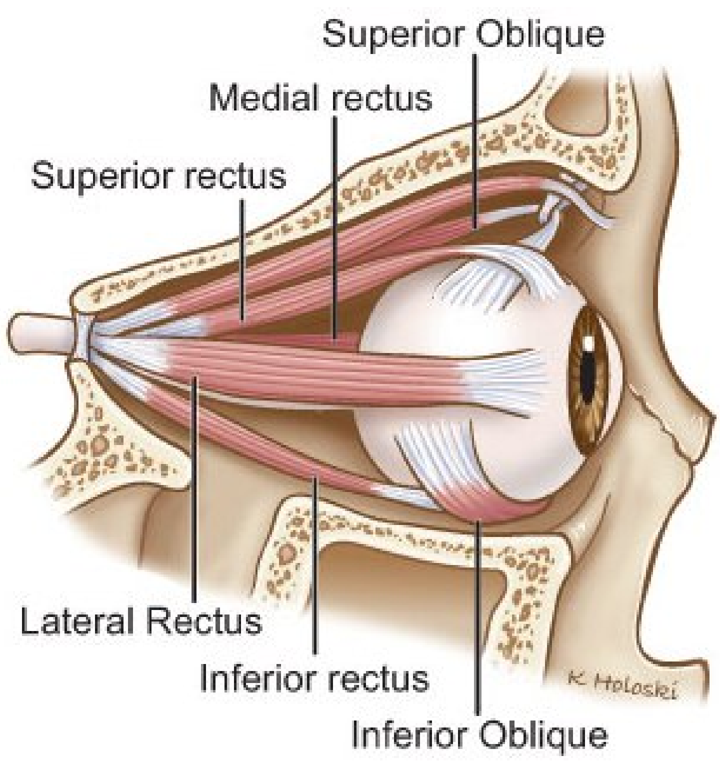


Figure 3.1: The Six Extraocular Muscles in a Human Eye Socket.

sensors (rotational and linear VOR) as shown in Figure 3.2 [95]. The neural control pathway of VOR is short and fast, with a latency of approximately 10 ms. However, it has been shown that head motions do not only cause neurally-controlled VOR. In humans, the VOR gain following passive head oscillation increases with increasing head frequency and passes unity at approximately 12 Hz. It was speculated that this high gain was caused by mechanical oscillations of the orbital apparatus [96]. When a rhesus monkey was accelerated laterally on a sled for 200 ms with peak acceleration of approximately 400 mm/s^2, an "anti-compensatory" eye motion with a duration of less than 40 ms and an amplitude less than 10% of the maximum compensatory response was found prior to the onset of the compensatory linear VOR [97]. When a monkey was subject to a sudden brief period of free-fall, equivalent to a 1 g vertical linear acceleration, the earliest ocular response was an anti-compensatory downward eye movement, followed by an upward compensatory VOR [94]. While the gain of the linear

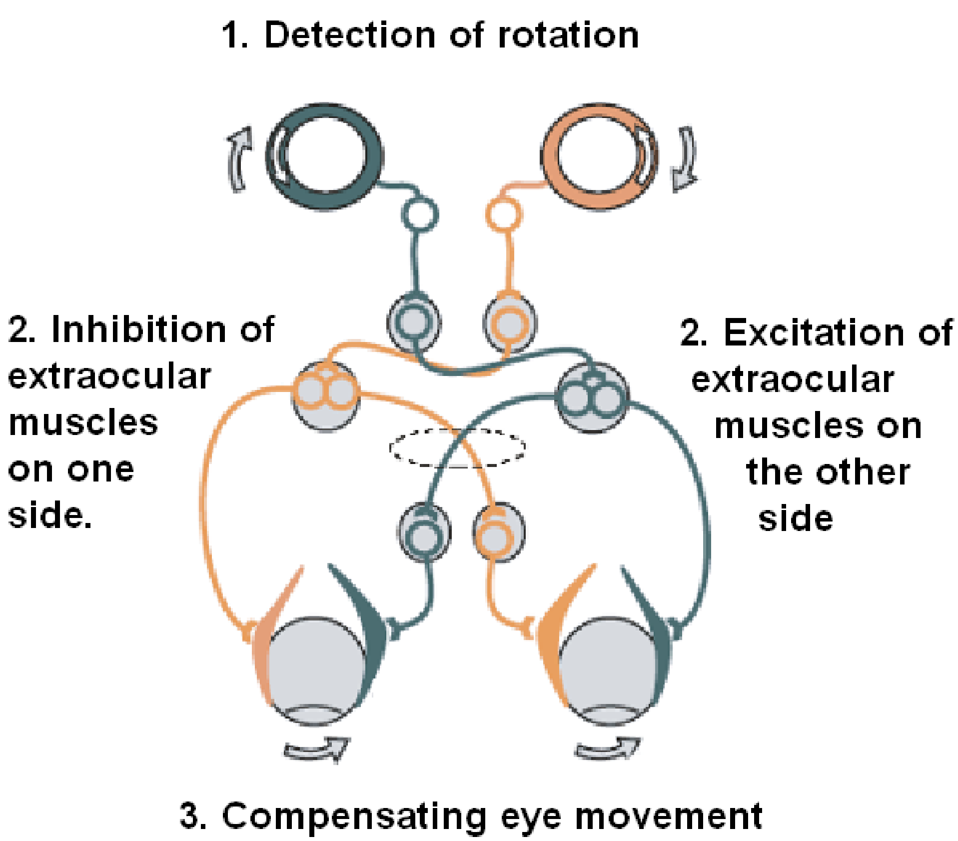


Figure 3.2: Vestibulo-ocular response (VOR).

VOR was sensitive to viewing distance, the early response was independent of where the animal was looking before the sled movement [94,97]. It was speculated that this early response was mechanical in nature and purely passive, i.e., not neurally controlled. When 200 ms rotational acceleration pulses (peaked at 1000-1200 deg/s^2 in 10 ms) were applied to the head of a human volunteer using a torque helmet, a zero-delay anti-compensatory eye response that preceded the compensatory VOR was found [21]. The anti-compensatory response was in the same direction as the head, had a peak velocity of several deg/s, a peak acceleration of several hundreds of deg/s^2 and amplitude of a few minutes of arc as illuminated in Figure 3.3 [21], where eye velocity shown inverted for clarity. More prominent anti-compensatory eye responses were found in patients with bilateral labyrinthine defects, presumably due to the disabled compensatory VOR [98,99]. Tapping the head with a reflex hammer produced a small, zero-delay passive response in electro-oculography that culminated in the first 5 ms [100]. These studies demonstrated that immediately after a HACC, the eye undergoes a passive response which appears to be purely mechanical. This indicates the eye, in its passive response, as an ideal candidate of brain surrogate.

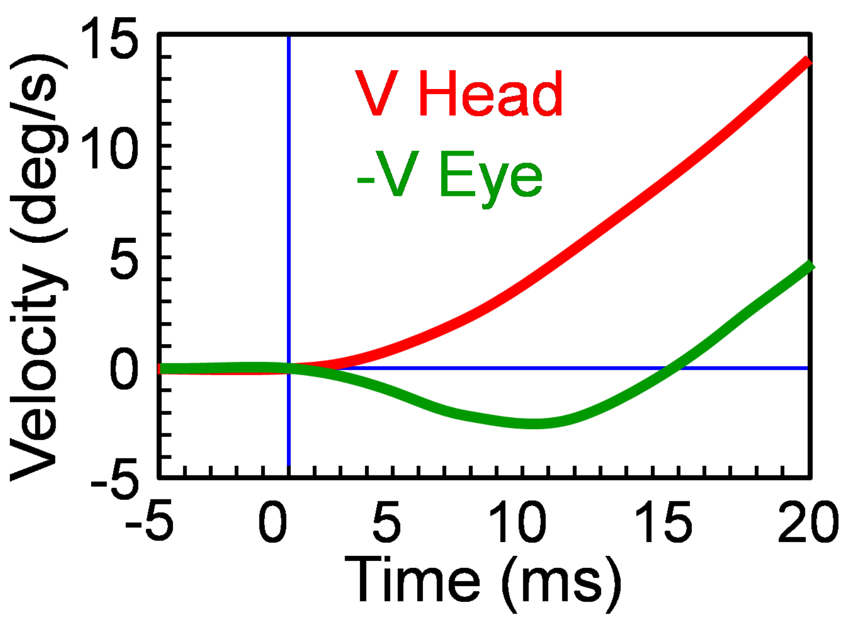
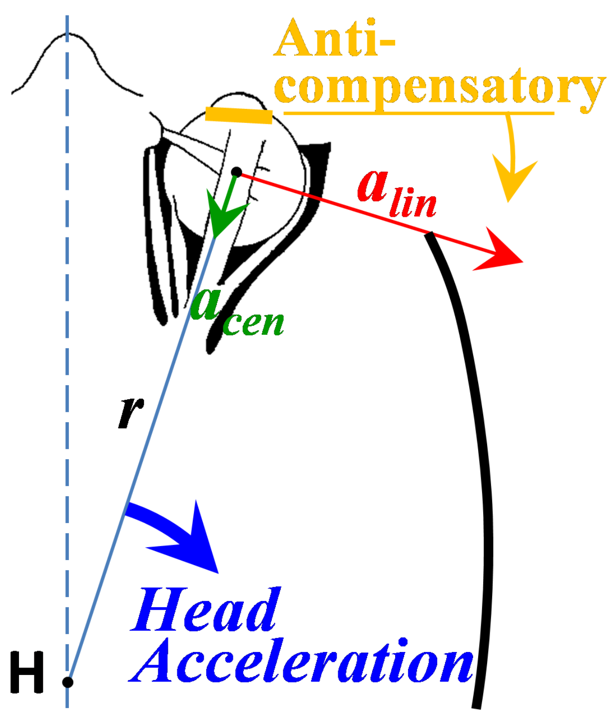


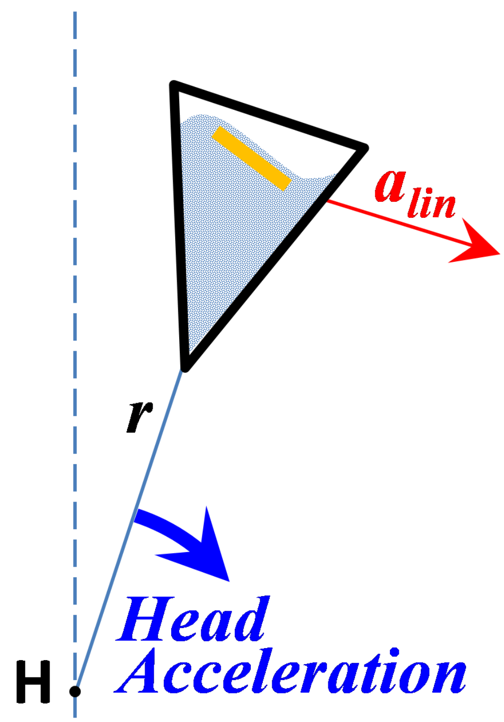
Figure 3.3: Time Courses of Head and Negative Eye Velocities.

It has been argued that the passive eye response (PER) was not the eye rotating about its center of rotation due to inertia, because such a rotation would have been in the compensatory direction. To explain the anti-compensatory PER, Collewijn et al. considered the eye and the surrounding tissues, the extraocular muscles, the orbital tissue and so on, as a viscoelastic mass (ocular mass) inside a bony confinement, r mm from the instantaneous center of rotation of the head (point H in Figure 3.4), that underwent a passive underdamped eccentric rotation during HACC [21]. The tangential component of the rotational moment, a_{lin}, deformed the ocular mass such that its front surface tilted in the same direction as the HACC (Figure 3.4b). This deformation was registered by the search coil fixed on the front surface of the eye as an anti-compensatory eye rotation. Using semi-solid gelatin to simulate the eye mass, Collewijn et al. successfully simulated the first order effect of the PER [21]. They concluded that "the anti-compensatory component can only be purely mechanical in origin and can only result from the assembly of eye accelerations (rotational and linear) related to the imposed head rotation." [21]. When the PER and the active VOR were separated mathematically, it was found that, without active VOR, the eye would have undergone a damped 12-15 Hz oscillation, starting in the anti-compensatory direction [21]. This VOR-truncated passive oscillation of the eye at its natural frequency can also explain the results from squirrel monkey VOR studies in [101] and the anti-compensatory eye responses under linear accelerations of the head in [94] and [97].

Therefore it is important to distinguish the normal, neurally-controlled eye movement, which is a rotation around the center of rotation of the eye, from the PER, which is the displacement and deformation of the ocular mass in the eye socket under the impact load of the head. It is also important to point out that, while the PER reported in the studies mentioned above was recorded as a rotation of an eye coil, its underlying mechanics resembles that of the initial phase of "liquid sloshing" (Figure 3.4). This sloshing can be easily observed by pushing a bottle of liquid soap or a cup of coffee across the table and paying attention to the direction of the initial motion of the surface [102,103]. Sloshing of the ocular mass in the socket (displacement and deformation) was demonstrated using a simplified finite element eye-in-socket model when a fall from a 1-m height was simulated [104]. In hydrodynamics, sloshing is quantified by a velocity potential field, or a time-varying deformation throughout the body of mass that sloshes. For example, the surface layers of the body are known to move more freely than deeper layers [102, 105]. Studies of gelatin brains and human cadavers have shown sloshing-like deformations of the brain, with the surface undergoing larger motion than deeper layers and thus suffering stronger strain [45, 46, 106]. In fact, sloshing, in its hydrodynamic sense, has been used to describe the injury mechanism of the semisolid mammalian brain [107, 108] and "brain sloshing" is often found in the press and on TV. This has led to the development of a physical antisloshing method that resulted in a significant reduction of the pathological index of TBI in a rat model [107, 108] and to an explanation of the lower concussion rates in NFL games played at higher altitudes [109].



(a)



(b)

Figure 3.4: Illustration of Eye Acceleration and Sloshing: (a) Eccentric Rotation of the Human Eye; (b) Sloshing of the Ocular Mass Due to Head Acceleration.

Another line of research also shows that brain and eye injuries are closely related. It was found that 78% of children with abusive head trauma (shaken-baby syndrome) had retinal hemorrhages [110] and 66% of US service members with TBI also had combat ocular trauma [111]. It is thus hypothesized that a head/body impact causes both the ocular mass and the brain to undergo a sloshing-like motion and deformation and that direct assessment of PER immediately after the impact, in combination with assessment of the head kinematics, can be used to infer brain response and to better assess the risk of mTBI.

Chapter 4

Sensing Brain Response to Linear Acceleration of the Head

In this chapter, an alternative approach to sense and record the passive eye response as a surrogate for the brain response to the linear acceleration of the head is introduced. As discussed in the previous chapter, when an impact happens, the muscles that control the eyeball are unable to react for a time of approximately 20 ms [21]. Within this time window, the eye moves passively in a fashion similar to the brain, yet different from the head/skull. In this chapter, MEMS accelerometers are first validated to sense and record the different movements of the model eyeballs and model eye sockets in hit-by-hammer experiments. The sensing technique is then evolved by using MEMS inertial measurement units (IMU) with sensor fusion to fit more dynamic application systems like the drop-and-impact experiments. Tests are performed on both 3D printed models and on human subjects. Relative linear velocities and displacements are calculated by processing the acceleration data. A semi-wireless in-helmet expansion of the skull model apparatus is designed by cooperating with colleagues from another research project.

4.1 Experiments with MEMS Accelerometers

4.1.1 Tests on 5:3 Ping Pong Ball Model

As a first attempt to verify the eye as a surrogate of brain, and the usage of MEMS sensors to sense the passive eye accelerations, a series of human body phantoms with 3D-printed "bones" and gelatin emulated "tissues" were designed and tested. A 5:3 expanded eye socket model with a ping pang ball "eyeball" was initially tested. In order to emulate the relatively high tensile strength of the eye sclera, a model eyeball was fabricated by injecting a 10% gelatin type A (Lot No. Q5860, Cat No. 901771) aqueous solution into a 40 mm diameter ping pong ball. As a real human eyeball has a diameter of approximately 24 mm, this model has an expanding ratio of 5:3. In this ratio, an eye socket was designed using SolidWorks™ CAD software according to human anatomy shown in Figure 4.1 [112] but idealized in symmetry as shown in Figure 4.2a and Figure 4.2b, and 3D printed (Figure 4.2d) using polylactide (PLA) with layer height of 0.1 mm and infill of 70% by a MakerBot™ Replicator. Notice that in Figure 4.2d, the ping pong ball was suspended in cured 10% gelatin and covered by a 1-mm-thick "eyelid" membrane which was made of polydimethylsiloxane (PDMS). A pair of 3D-printed PLA molds (Figure 4.2c) were used to form the PDMS "eyelid". A 5:3-expanded cavity between the "eyeball" and the "eyelid" was specifically spared as a control according to [113]. The PDMS in the molds had a 10:1 ratio of elastomer to curing agent. After removing bubbles by placing the well mixed PDMS in a vacuum chamber, the PDMS mixture was poured into the molds and baked in an oven at a temperature of 70 °C for 6 hours. A pair of PLA clamps were also 3D printed to fasten the model during impact tests (Figure 4.2d).

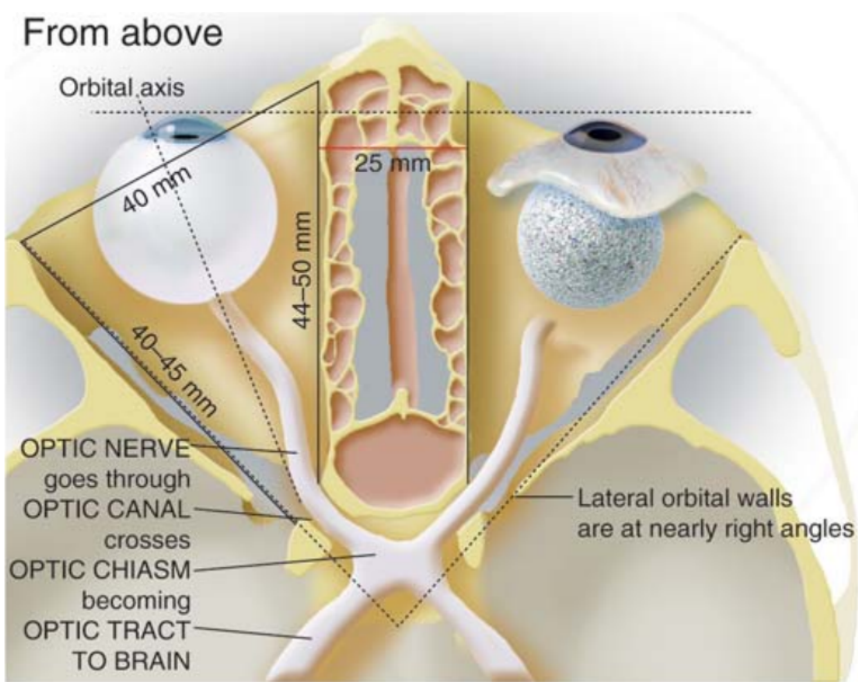


Figure 4.1: Anatomic Dimensions of Human Eye Socket.

Analog Devices™ ADXL325 3-axis MEMS accelerometers were used for the impact tests [114]. The ADXL325 is a small, low power (single supply operation at 1.8 V to 3.6 V), complete 3-axis (X, Y, and Z) accelerometer with signal conditioned voltage outputs. The output signals are analog voltages that are proportional to acceleration. One of these 4

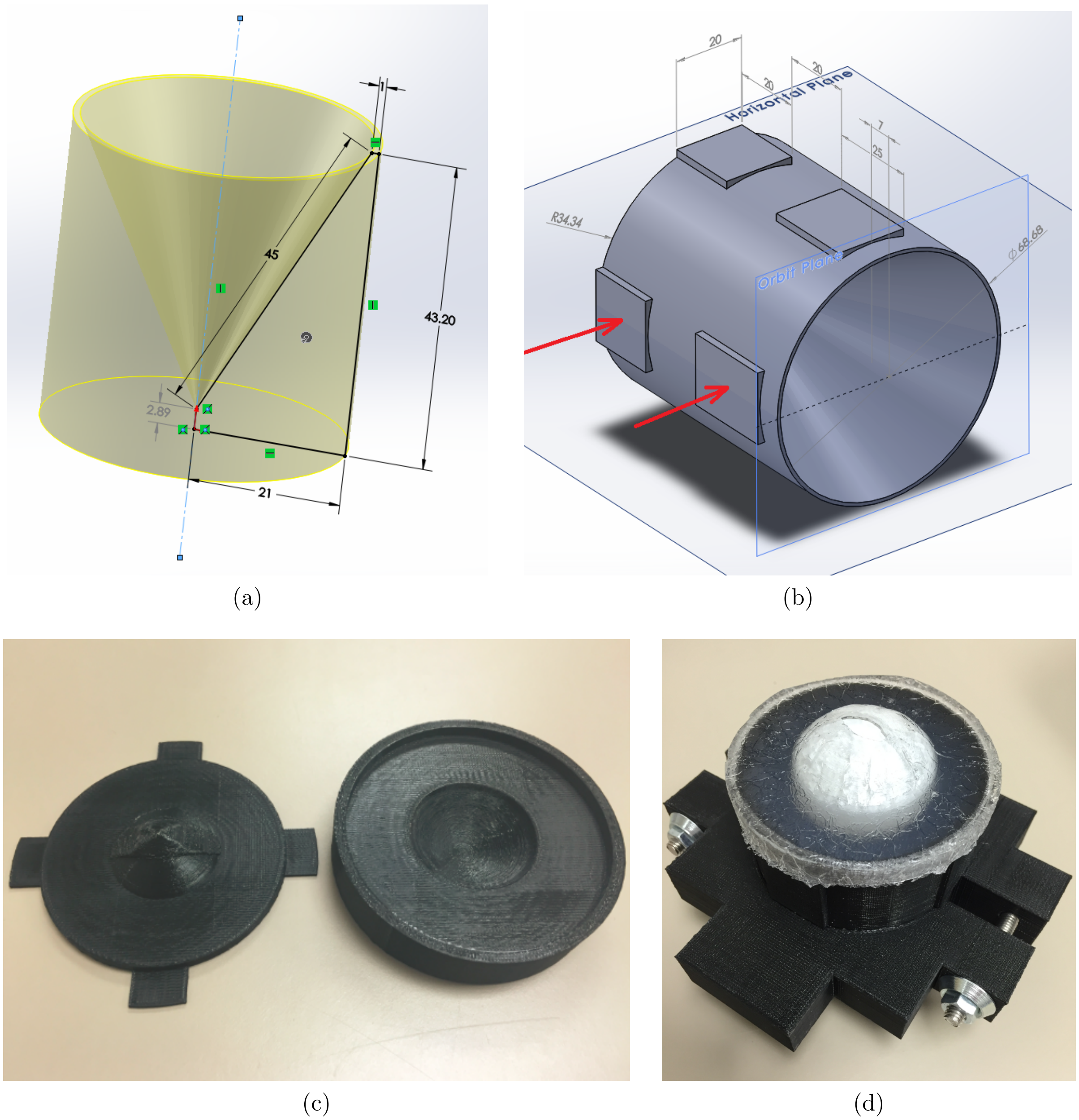
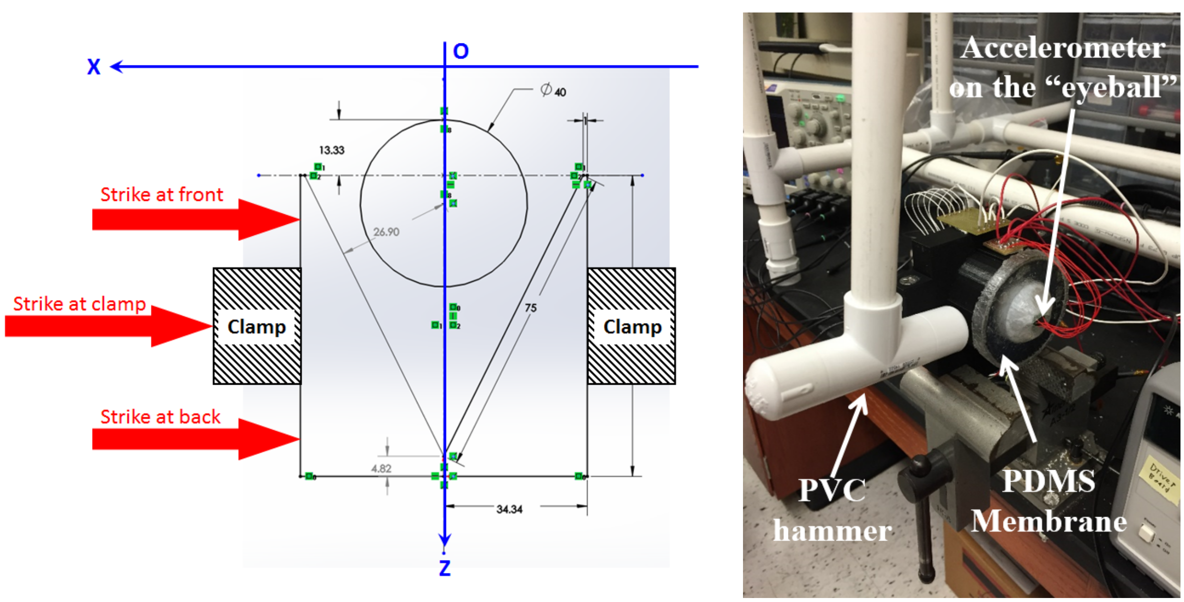


Figure 4.2: 5:3 Eye Socket Model: (a) SolidWorks™ Design with Dimensions in mm; (b) 3D Model with Impact Platforms and Directions of Impact (in red); (c) 3D-printed Molds to Make PDMS Eyelid; (d) The Setup of the Model.

mm × 4 mm × 1.45 mm leadframe-based chip-scale packages (LFCSP) was mounted on a 8 mm × 6 mm × 0.79 mm customized printed circuit board (PCB) (Figure A.1) and which was connected via AWG 32 wires. It was inserted in the PDMS "eyelid" membrane cavity and held in position on the front side of the ball by the membrane. A second ADXL325 accelerometer, as a reference sensor coupled to the motion of the socket, came on a 20 mm × 20 mm × 1.57 mm evaluation PCB labeled as EVAL-ADXL325Z, and was attached on the backside of the socket model so that the Z-axes of both accelerometers were aligned. The X-axes of the two accelerometers were parallel and along the directions of the strikes applied on the socket model (Figure 4.3a).

Since the demodulator output designed on the chip of ADXL325 is amplified and brought off-chip through a built-in 32 kΩ resistor, the user sets the signal bandwidth of the device by adding a capacitor to each output, and the bandwidth is then calculated by the 3 dB bandwidth equation of Equation (4.1) [114].

$$f_{-3dB} = 1/(2\pi \times (32k\Omega) \times C_{X,Y,Z}) \tag{4.1}$$



(a) (b)

Figure 4.3: Illustration of Impact Tests: (a) Cross Section View; (b) Test Apparatus.

where $C_{X,Y,Z}$ is the filter capacitor at each axis output. This filtering improves measurement resolution and helps prevent aliasing. In this application, the output of each axis of the accelerometers was filtered by a 0.1 μF capacitor on a separate interface circuit board (Figure A.3) fabricated by a LPKF ProtoMat S103 circuit board plotter by LPKF Lasers and Electronics^{TM} [115], resulting in a bandwidth of 50 Hz. The noise of ADXL325 has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of μg/$\sqrt{Hz}$, i.e., the noise is proportional to the square root of the bandwidth. Therefore, the bandwidth had to be limited to the lowest frequency needed by the application so as to maximize the dynamic range and the resolution (minimum detectable acceleration) of the sensor. This traded-off bandwidth of 50 Hz was just wide enough for the the natural frequency of 12-15 Hz for the VOR-truncated passive oscillation of the eye [21] as discussed in Section 3.3. This design was also verified by the spectra of the later processed acceleration data. Under the conditions of a supply voltage of 3 V, filter capacitors of 0.1 μF for all three axes, and a room temperature of 25 °C, the ADXL325 achieved a sensing range of ±5 g, a typical sensitivity of 174 mV/g, and a typical noise density of 250 μg/$\sqrt{Hz}$.

The test apparatus was a 30 in × 20 in PVC pipe frame holding a hammer made of a PVC T-shape pipe joint with a lever arm of 20 in. The other end of the arm was suspended at the top of the frame in such a way to make the hammer swing freely in the direction orthogonal to the frames long axis. During the testing, the hammer was lifted half way in the air and released to generate an approximately 5 g strike on the socket model, emulating the impact applied to the head and then translated to the bony eye socket. As illustrated in Figure 4.2 and Figure 4.3, the model was designed in a way that strikes can be applied at three points: the front platform, the clamp and the back platform. Since the direction of the strikes were parallel with the X axes of the accelerometers, the data from these outputs were of greater interest.

The data of acceleration from the two accelerometers were captured and recorded simultaneously by two Tektronix^{TM} oscilloscopes (MDO4054 and MDO3104) with the same

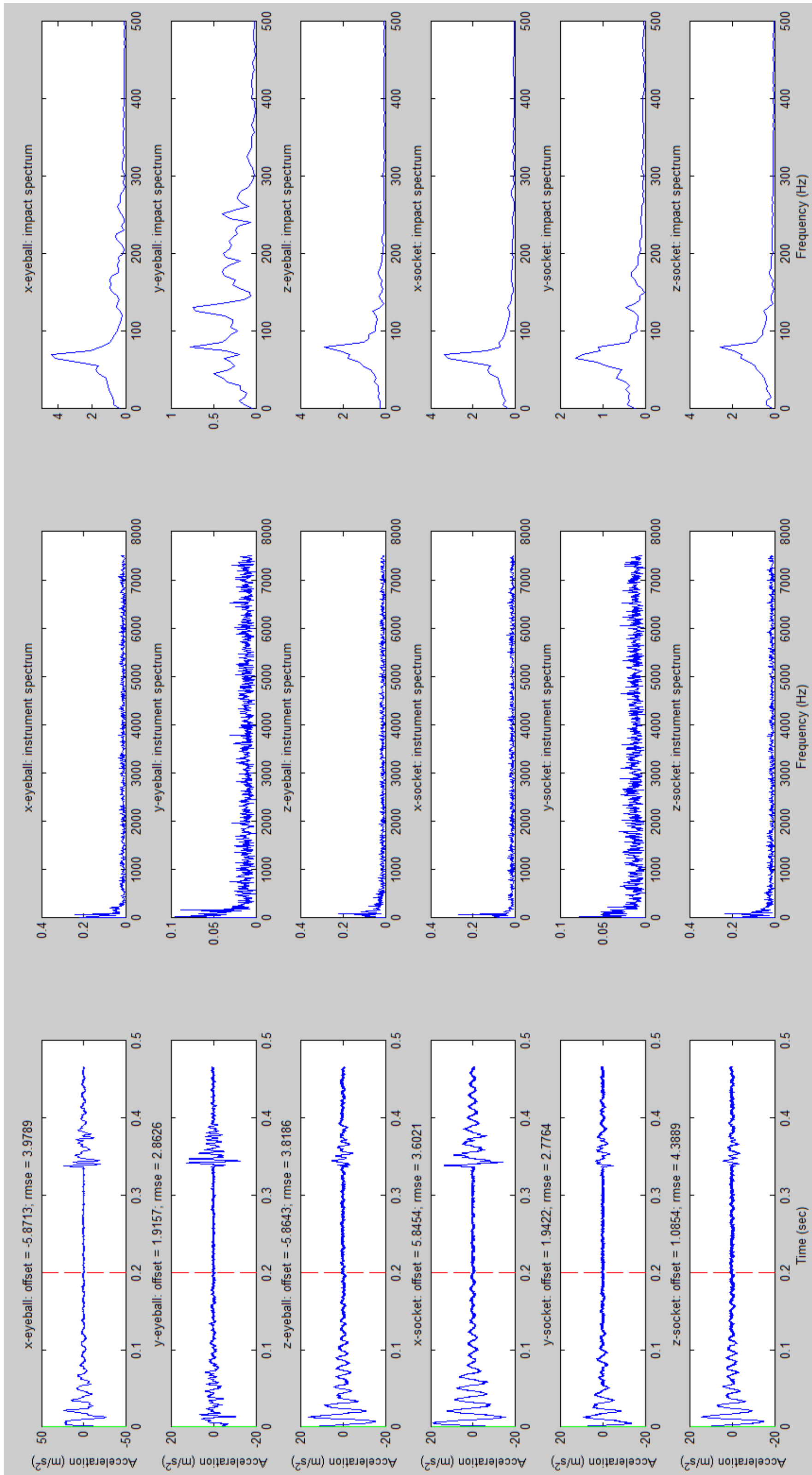


Figure 4.4: Raw Data of a Typical Impact Test with the Spectra of Instrument and Impact.

sampling rate of 50 kHz per channel. Data were stored on USB flash drives and processed by a MATLAB program (Appendix B.2) on a desktop workstation. The raw data with spectra of a typical impact test on the front platform are shown in Figure 4.4. The raw data of the accelerations output by the X, Y and Z axes of the eye accelerometer and the head accelerometer are in the first column with solid green vertical lines marking the starting time of the first/major impact and dashed red veritical lines marking the end of impact. Fast Fourier transform (FFT) was applied on this section of data resulting in the impact spectra shown in the third column of Figure 4.4. Before the impact, a backgound signal was also captured and transformed by FFT, resulting in the instrument spectra shown in the second column of Figure 4.4. Notice that the X axes of both accelerometers output the largest accelerations in magnitude, which corresponded with the impact direction in X. In the impact spectra of both accelerometers, the peak frequency components have bandwidths smaller than 50 Hz, which confirm the previous hardware design (except for the Y axis of the eyeball sensor, which was of least interest in this design of experiment).

A band pass filter (BPF) with low cut-off frequency of 25 Hz and high cut-off frequency of 500 Hz was used to filter the raw data which was then processed to calculate the velocities

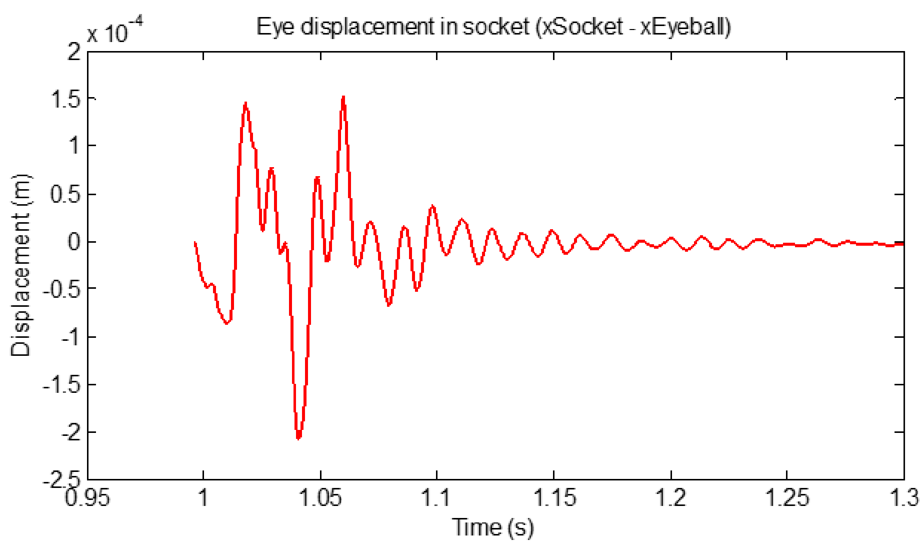
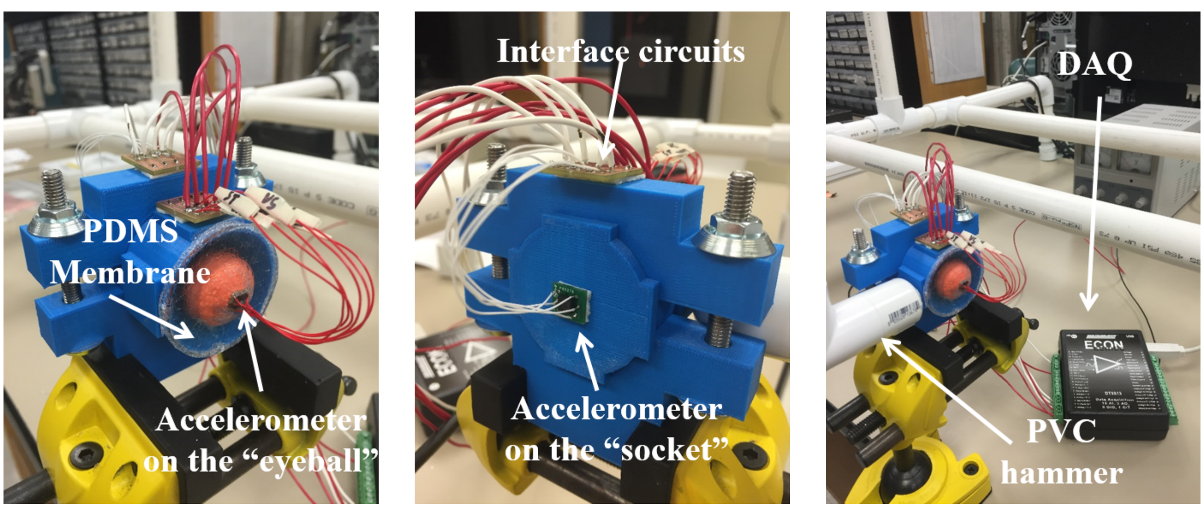


Figure 4.5: Relative Displacement of the Eyeball Model in the Socket Model along X-axis in the 5:3 Model Set.

and displacements of the eyeball and socket models by applying cumulative trapezoidal numerical integration [116]. Figure 4.5 is a plot of a typical result of a calculated relative displacement of the eyeball model in the socket model. The plot started at the time when the impact happened and damped to about 0 within 0.3 s. The oscillation with a maximum amplitude of 0.21 mm indicated that the eyeball model moved differently compared with the socket model. The very slight shift from 0 after converging shows that the gelatin, which was used to simulate the muscles holding the eyeball, deformed during the impact.

4.1.2 Tests on 1:1 Rubber Ball Model

To diminish the influence on the results from the size, a scaled eyeball and socket model set was made. In this version, the "eyeball" was a rubber ball with diameter of 1 in or approximately 25.4 mm, which is very close to the average diameter of human eyeball. The design of the socket was generally identical to the 5:3 version except for being smaller in size, and, because of that, only the strike platforms at the front were spared with the clamp, as shown in Figure 4.6. A flexible PCB was fabricated to minimize the size of the sensor



(a) (b) (c)

Figure 4.6: Test Apparatus for the 1:1 Scaled Model.

assembly (Figure A.2). This allowed for much easier insertion into the 1:1 scaled model membrane.

Instead of using two oscilloscopes, a Data Translation^{TM} DT9813 USB data acquisition (DAQ) system was used to collect data. A MATLAB controlled software of QuickDAQ^{TM} was used for processing the data (see code in Appendix B.1). The sampling rate was 16.67 kHz per channel. Figure 4.7 plots a typical calculated relative displacement of the scaled eye model to the socket model. Notice that, compared with the 5:3 expanded version, the maximum amplitude of oscillation was smaller and the damping was faster (within 0.15 s). This is due to the smaller socket space of the scaled model. The test results from both versions of eye socket model confirmed that, under an impact of approximately 5 g loaded on the socket, the eyeball moves differently and has a linear displacement relative to the socket/skull.

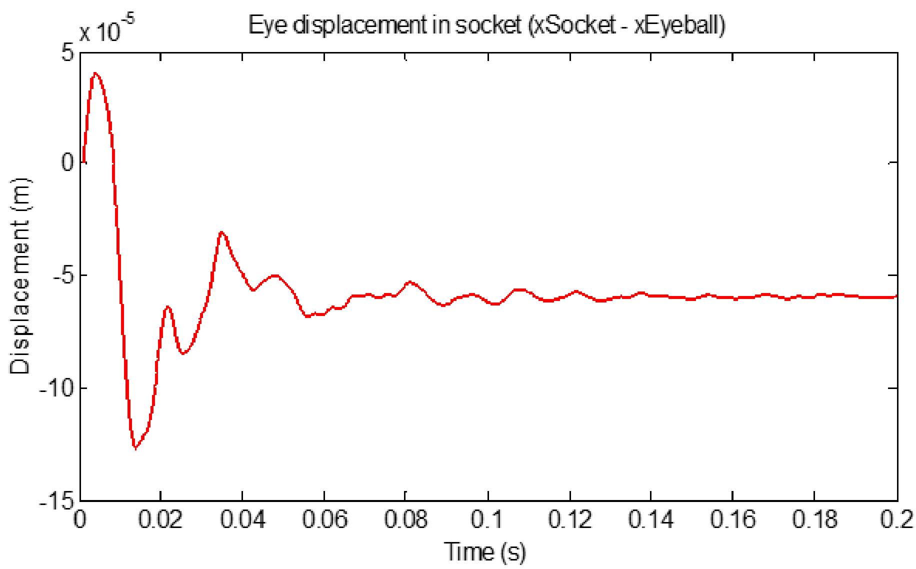


Figure 4.7: Relative Displacement of the Eyeball Model in the Socket Model along X-axis in the 1:1 Scaled Model Set.

4.1.3 Tests on Human Volunteers

The experiments on the two models of eyeball and socket provided clear proof that the eyeball moves differently than the skull does during an impact of approximately 5 g and the accelerometers were able to sense and record it simultaneously. In order to validate the accelerometers' ability to sense and record the passive movements of real human eyes, more tests were done on human volunteers. (The study was approved by the Institutional Review Board of UAB and conducted in accordance with the Declaration of Helsinki.)

As shown in Figure 4.8, the ADXL325 accelerometer packages are mounted on polyimide flexible PCBs designed in a ribbon style (layout in Figure A.2). The flexible sensor boards were spray coated with an acrylic conformal coating to block moisture and then spin coated with a 5:1 ratio PDMS film to make them biologically compatible. After being sterilized in Peroxiclear^{TM} solution (Figure 4.8b), the sensor boards were inserted in the volunteers eye sockets beneath the corneas. Before the tests, the volunteers lay on the leveled experiment bench with their heads resting on a support that was 2.8 cm in height. During the tests, the support was pulled aside quickly to let the head drop and the back of the head hit the bench surface. The data was captured by the data acquisition system described in the previous section. The sampling rate was 33.33 kHz per channel.

Since it was very difficult to tell the position and orientation of a flexible sensor board when inserted in a volunteers eye, the positive direction of the overall acceleration sensed from the removal of the head support to the end of impact was assumed to be opposite to the gravitational acceleration. The vector sum of the three axes was used to calculate the amplitude of the overall acceleration. Figure 4.9 plots a typical set of overall acceleration, velocity and displacement. Notice that there are two spikes on acceleration indicating that the head bounced off of the bench and hit again after the first impact. With the green vertical line marking the starting point of the first impact, one can clearly see that the sensor in the eye started to fall 0.176 s before the impact, moved upward shortly after the head impacted and bounced from the bench. The lowest point (2.863, -0.02782) of the displacement plot

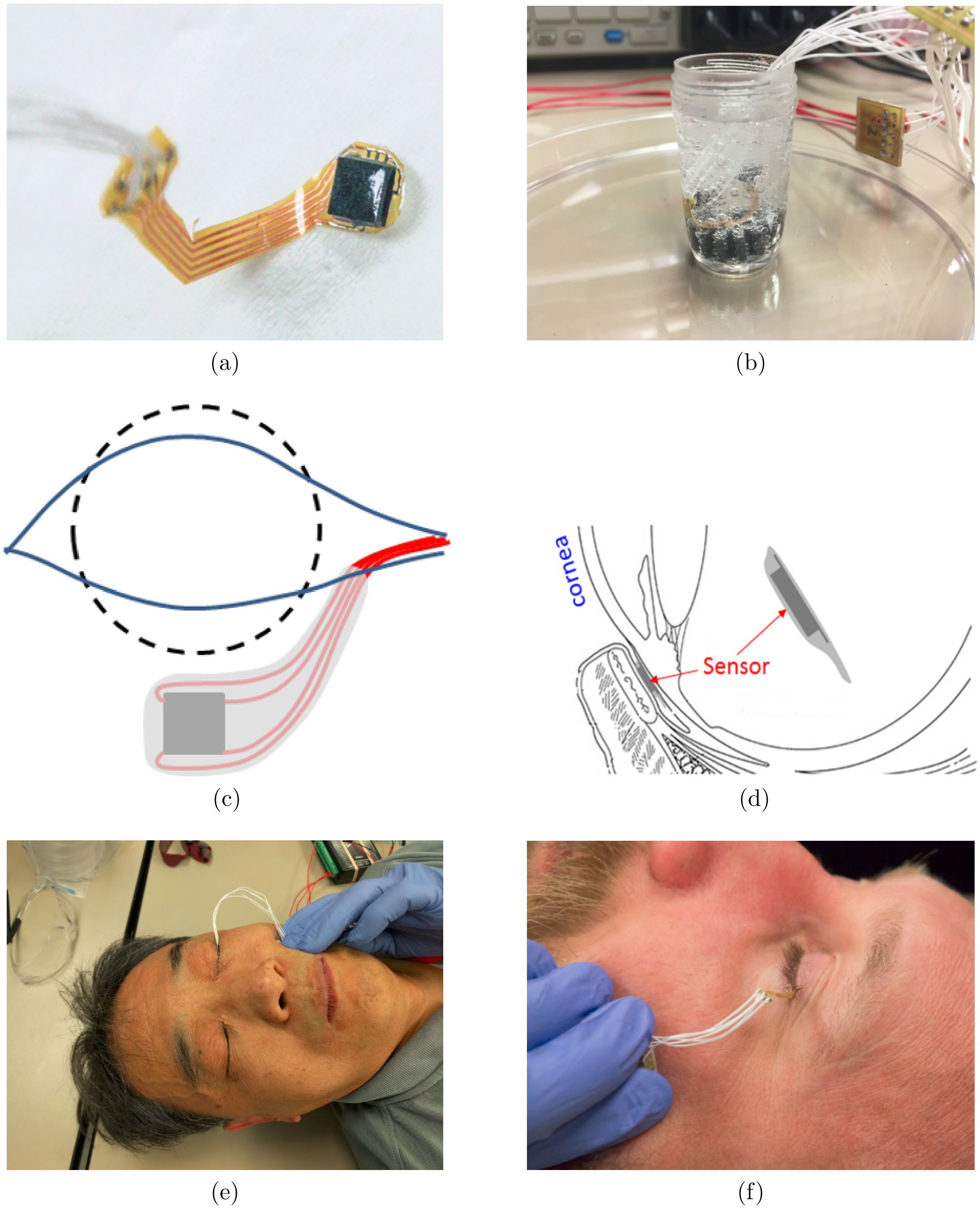


Figure 4.8: Illustration of Human Tests: (a) A Flexile Ribbon Style Sensor Board; (b) A Sensor being Sterilized; (c) A Front View of Eye Insertion; (d) A Side View of Eye Insertion; (e) A Photograph of a Volunteer (Dr. Lei Liu from UAB) with Sensor Inserted in the Lower Eyelid; (f) A Photograph of a Volunteer (Dr. Mark Bolding from UAB) with Sensor Inserted in the Lower Eyelid.

marked by the data cursor shows a delay from the impact time which verifies the difference in movement between the eyeball and the head. The calculated dropping distance of 2.798 cm accurately fits the actual height of the head support.

As a brief summary of this section, the differences in displacement of "eyeball" and "socket" were shown by the data from both models. It indicated that the MEMS accelerometers have potential to sense and record the PER as a surrogate for brain response to head acceleration during a concussion causing impact. Human tests also validated the use of MEMS accelerometers. The results of this section were presented at the IEEE SENSORS 2016 conference and published in [117].

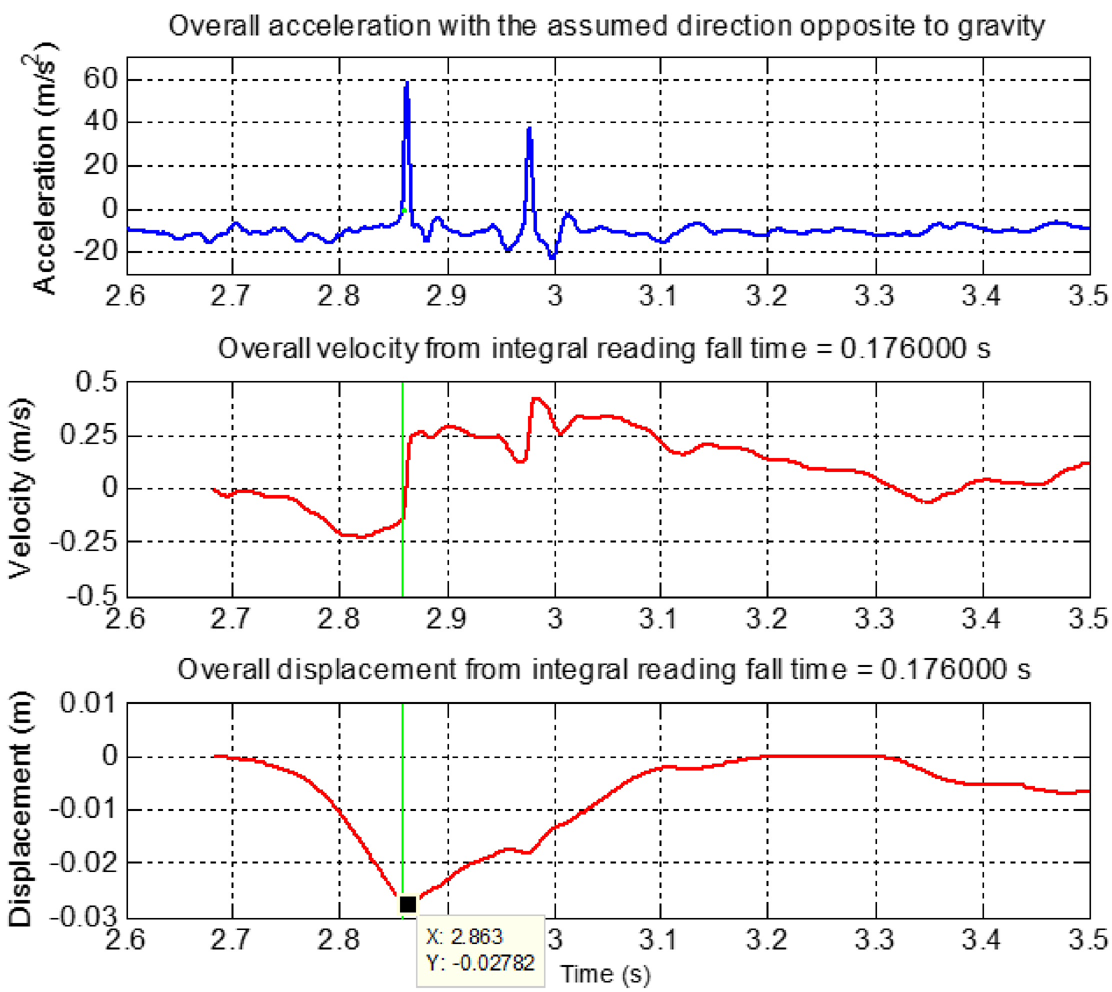


Figure 4.9: A Typical Result from the Human Tests with Accelerometers.

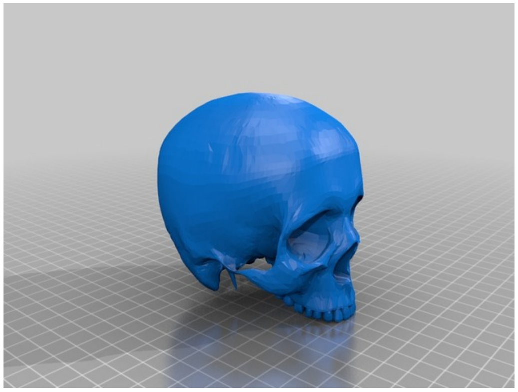
4.2 Experiments with MEMS IMU's

As confirmed previously, MEMS accelerometers are able to measure the relative displacements of eye and head caused by impact when the displacement of the entire system is negligible due to their vibrational sensitivity. In a dynamic system, which is closer to real-world application, the knowledge of the sensors' orientation is necessary for accurate estimation of the measurands. Therefore, in this section, a dynamic testing apparatus containing a scaled skull phantom and scaled eye replicas were designed and tested with MEMS IMU's, each of which combined a 3-degree-of-freedom (3-DoF) accelerometer and a 3-DoF gyroscope. An advanced sensor fusion technique was developed to produce accurate estimation of linear accelerations and displacements of the eye relative to the skull under impacts.

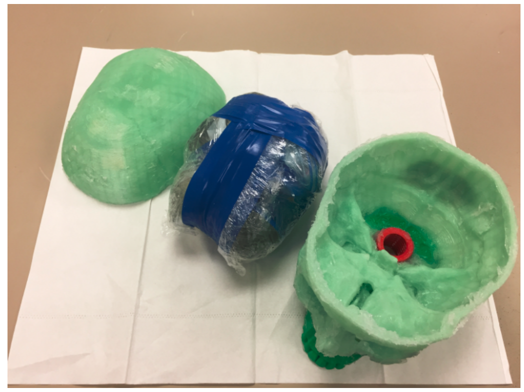
4.2.1 Model Design and Data Acquisition

The skull model used for testing was 3D printed using PLA with a MakerBot^{TM} Replicator [118]. To better approximate a human skull, a model gelatin brain was constructed using a ratio of 42.86 g of gelatin per liter of water and encased in the skull (Figure 4.10 [118]). Eye replicas were fabricated using a 10:1 PDMS:curing agent ratio cured at 40 °C overnight in a 3D-printed PLA mold. PDMS was selected for its similar density to water. This allowed the volume and mass of the replica eye to closely approximate that of a real eye. Replica eye diameter equaled 24 mm (Figure 4.11c), approximating actual eye diameter [119]. Additionally, the mass of eye replicas was about 7.3 g (Figure 4.11d); close to actual eye masses of about 7.5 g [120]. The eye sockets were filled with gelatin to emulate eye socket tissue and to hold the replica eyes in place. Previous work in [121] utilized gelatin to simulate the mechanical properties of the eye socket when subjected to a force. Once held in the gelatin sockets, replica eyes were covered with 1 mm thick PDMS films to act as eyelids. PDMS to curing agent ratio was 10:1.

Two MEMS LSM6DS3 IMU's by STMicroelectronics^{TM} [122], containing a 3D accelerometer and a 3D gyroscope, were used to capture the inertial measurands of the model

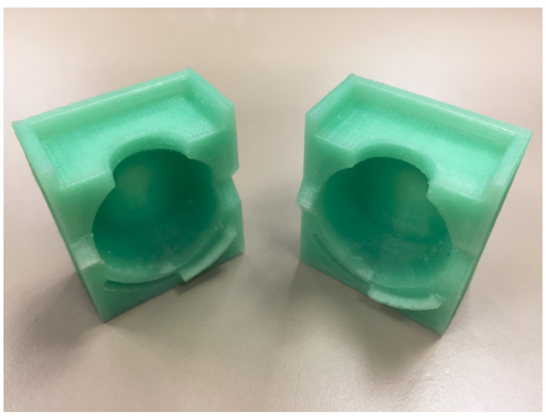


(a)

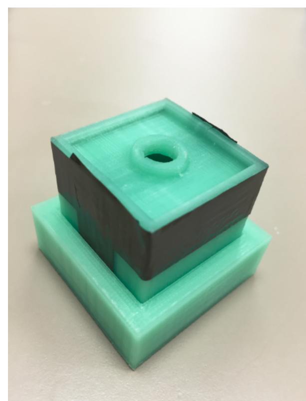


(b)

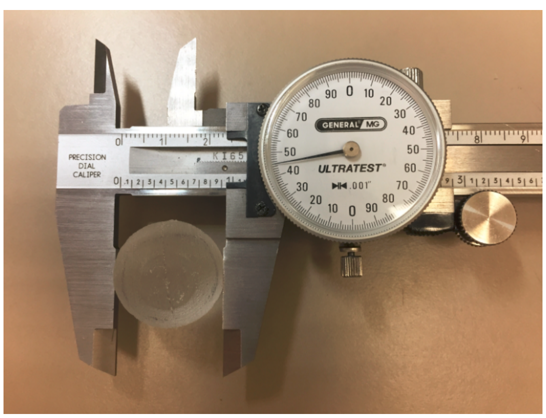
Figure 4.10: The Human Skull Model: (a) 3D CAD Model; (b) 3D-Printed PLA Skull Model and Gelatin Brain.



(a)



(b)



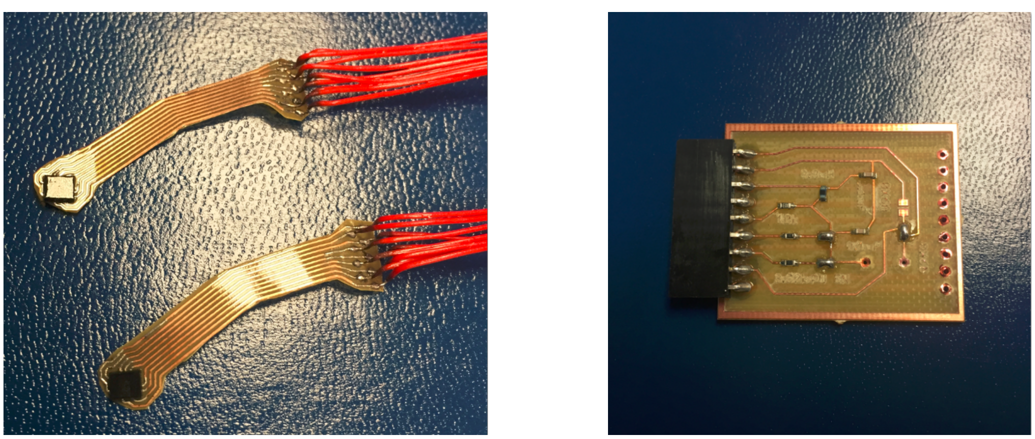
(c)

(d)

Figure 4.11: The Human Eye Model: (a) 3D-Printed Eye Molds; (b) Eye Mold Set; (c) PDMS Eye Model Measured by a Caliper; (d) PDMS Eye Model on a Balance.

set. An LSM6DS3 is packaged in a land grid array (LGA) package with dimensions of 2.5 mm × 3 mm × 0.83 mm. One of the IMU's was mounted on a flexible ribbon-style PCB made of DuPont™ Pyralux, and was then coated with silicone conformal coating (Figure 4.12). The conformal coating was done by spraying TechSpray™ silicone coating solution to the sensor PCB and then baking at 60 °C for 8 hours to accelerate the curing. The purposes of the coating include protecting the circuits from moisture, protecting human test volunteers from electricity, smoothing edges to avoid scratching, and making the surface biologically compatible. Biological compatibility is defined "the quality of not having toxic or injurious effects on biological systems" by Dorland's Medical Dictionary. Although the substrate material, polyimide [123], has an insignificant level of cytotoxicity [124], the off-the-shelf IMU packages, like the LGA in our case, contain multiple materials with unproven biological compatibility [125]. Although most of the soldering flux is removed, any remaining flux must also be encapsulated due to it's mild toxicity.

The completed skull model was attached to a PVC neck of 11 cm [126] which hinges at a 3D-printed pivot joint attached to a lab table (Figure 4.13a). This allowed the skull model to fall in an inverted pendulum orientation resulting in impact with the table. Drop tests consisted of elevating the skull model via a brass rod held by a multi-leveled 3D-printed



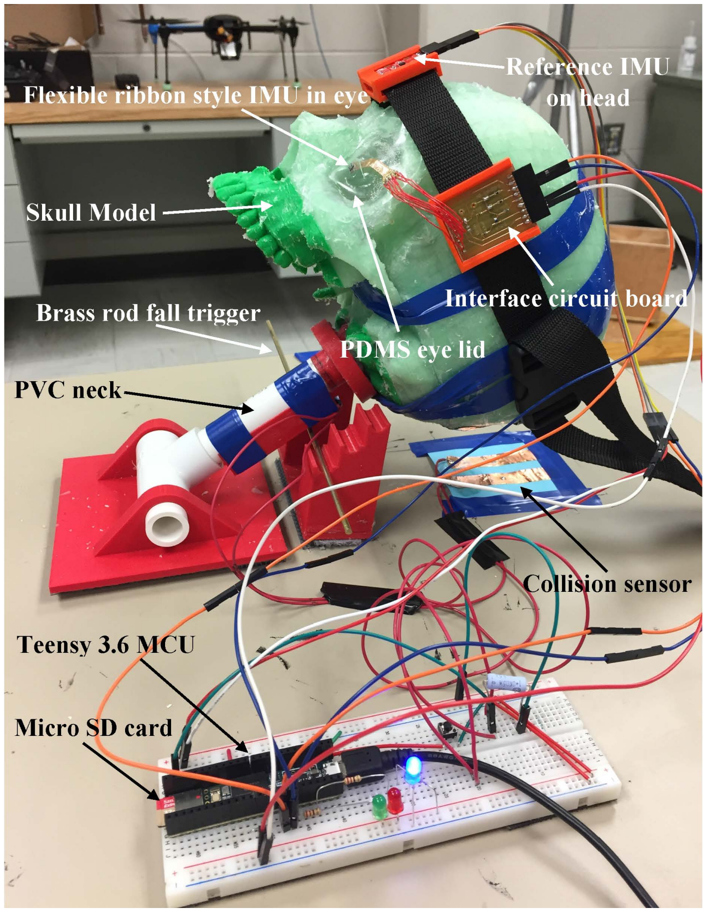
(a) (b)

Figure 4.12: The Sensor to be Inserted in the Eye: (a) Flexible Ribbon Style Sensor PCB; (b) Interface Circuit.

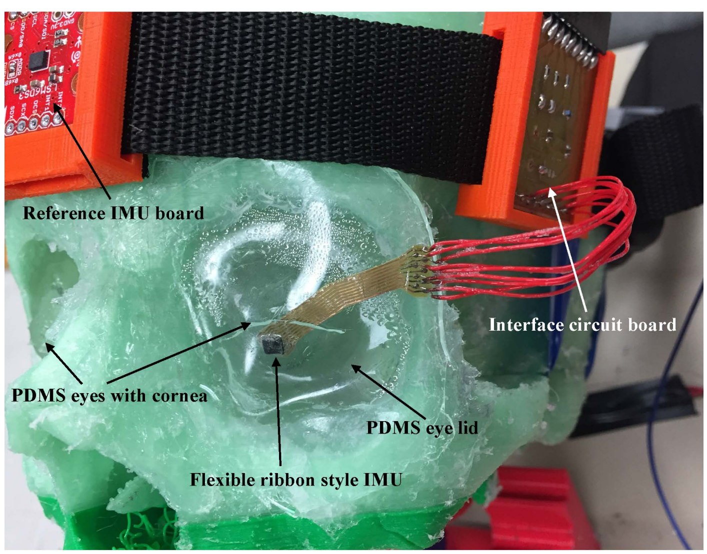
platform such that the base of the skull was 85 mm from the table. Drops were initiated by the removal of the brass rod supporting the skull, allowing the skull to fall and collide with the table.

The flexible ribbon style sensor was inserted into the eyelid model through the eyelid slot opening and held on the lower surface of the eye replicas by the lower eyelid (Figure 4.13b). The other IMU was mounted on a breakout board by SparkFun™ [127]. As a reference sensing set, this breakout board was held by a 3D-printed PCB holder that was tied firmly to the skull to eliminate relative movement to the skull. The x-axes of the two IMU's were adjusted, by tuning the PVC neck, the reference IMU holder and the flexible PCB to align the sensor edges to a reference horizon line on the test bench, such that they lie parallel to each other and perpendicular to gravity. Additional misalignment during drop tests was further compensated in the data processing algorithm introduced in Section 4.2.4.

A Teensy™ 3.6 microcontroller unit (MCU) [128] was used to control and communicate with the two IMU's via I^2C buses, acquire sensing data, and log the data in a micro SD card for further processing (Figure 4.13a). Through the Teensy 3.6 MCU, the accelerometers in the LSM6DS3 IMU's were programmed (see code in Appendix C) to operate with a sensing range of ± 16 g and a sensitivity of 0.448 mg/LSB (least significant bit) , while the gyroscopes operated with a sensing range of ± 2000 deg/s and a sensitivity of 70 mdps/LSB. The sampling rates and bandwidths of both the accelerometers and gyroscopes were set as 1.66 kHz and 400 Hz respectively to obtain the best performance of the IMU's. According to the data sheet of LSM6DS3 [122], the accelerometer has a maximum output data rate (ODR) of 6664 Hz and the gyroscope has a maximum ODR of 1666 Hz. We set the maximum ODR of the gyroscope as the sampling rate so as to synchronize the two sensors. The bandwidth is determined by the ODR selection suggested by the data sheet to set an anti-aliasing filter for high performance. All the sensing data, including temperature readings from the two IMU's, were buffered in the 256 KB random-access memory (RAM) of the Teensy 3.6 before



(a)



(b)

Figure 4.13: The Model Test Apparatus: (a) An Overview of the Test Apparatus; (b) A Zoom-in of the Eye.

being logged to the micro SD card. The overall sampling rate was 1 kHz which was sufficient for this application.

The brass rod supporting the skull also served as a trigger mechanism to signal the start of the model's fall. Connected to the PVC neck were two electrodes which, when both contacted the brass rod, allowed current to flow and registered a signal low. When the brass rod was removed and electrical contact with the electrodes was broken, the output signal switched to high, indicating the start of the model's fall. Upon impact with the table, copper conductive tape adhered to the base of the skull shorted the two conductive tape electrodes that were placed in close proximity on the table. This registered a signal high again and indicated the time of impact. This trigger signal was recorded and synchronized with the recorded sensor data (Figure 4.13a). A simple functional block diagram of the test apparatus is shown in Figure 4.14.

4.2.2 Tests on Skull Models

As confirmed in Section 4.1, accelerometers are able to measure the relative displacements of eye and head caused by impact when the displacement of the entire system is negligible due to their vibrational sensitivity. In a dynamic system, the knowledge of the sensors' orientation is necessary for accurate estimation of the measurands. In our experiment, specifically, the skull model experienced both linear and angular displacements from being supported to colliding with the test bench. The direction of impact, which was parallel

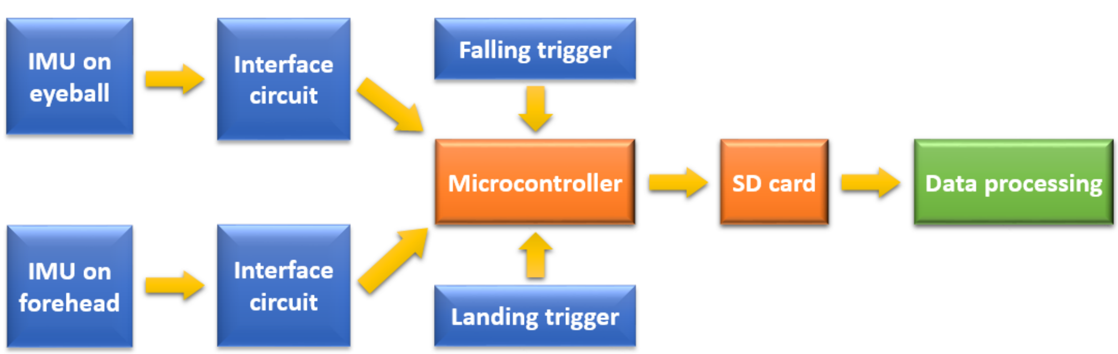


Figure 4.14: A System Functional Block Diagram of the Skull Model Test Apparatus.

to Earth's gravity and perpendicular to the test bench surface, was of most interest. In this direction the effect of the impact on the skull and the eye was studied as a simulation of a human head collision with the horizontal ground.

In this section, the orientation of the IMU along the x, y, and z axes are considered pitch, yaw, and roll respectively as shown in Figure 4.15. For each of the two IMU's in its local coordinate system, the positive x-axis extends horizontally from the right side to the left side of the skull model, the positive y-axis extends from the bottom to the top, and the z-axis extends from the back to the front. In this general positioning of the IMU's, we can collect the majority of the impact acceleration in the local z-axis and the rotational displacement in the local x-axis (pitch). The universal coordinate system is set so that the X-axis shares the same orientation of the sensors' local x-axes, the Z-axis is parallel to Earth's gravity vector with an opposite positive direction, and the Y-axis is determined by the right hand rule.

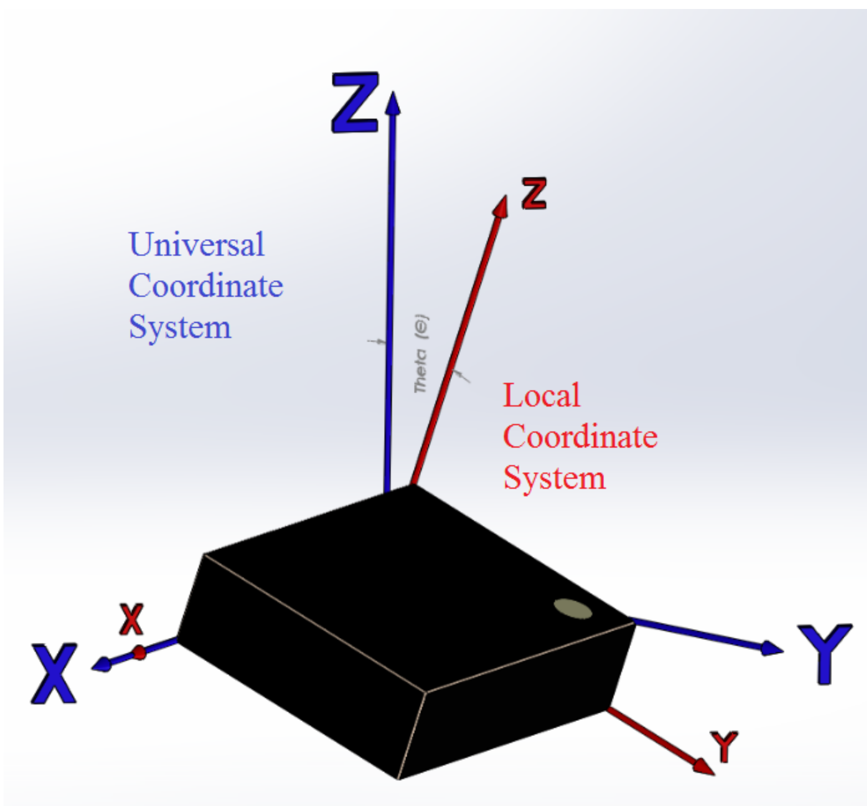


Figure 4.15: Local and Universal Coordinate Systems of an 6-DoF MEMS IMU.

The three axial accelerometers and the three axial gyroscopes of each IMU were used in the orientation estimation. While stationary, the accelerometer can be used to estimate the angle θ_A between the local and universal coordinate systems shown in Figure 4.15. Equation (4.2) was derived to compute the orientation with the acceleration data, where θ_A is the tilt angle between +z-axis and the universal +Z-axis, a_x, a_y and a_z are accelerations output by x, y and z sensing axes respectively, and $arctan()$ is inverse tangent function.

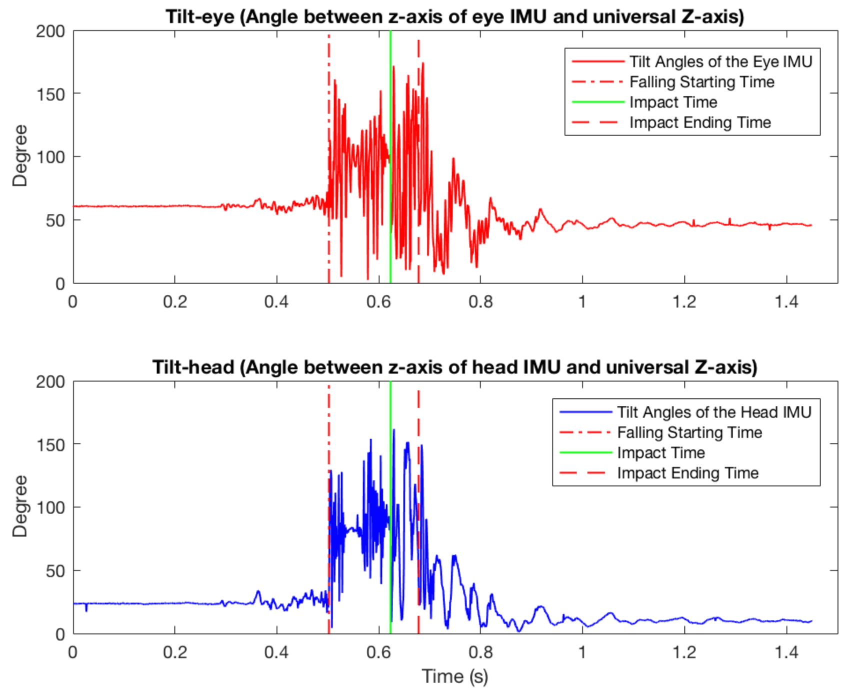
$$\theta_A = arctan\left(\frac{\sqrt{a_x^2 + a_y^2}}{a_z}\right) \tag{4.2}$$

The tilt angles of each IMU during suspension and after the collision were thus computed and used as the initial and final angular position to aid the rotation estimation by the gyroscope.

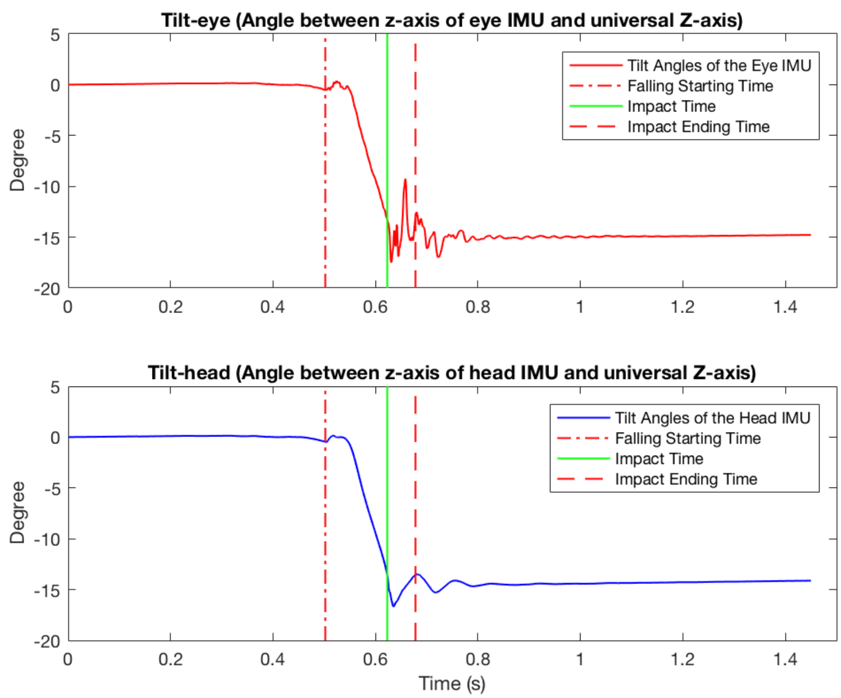
During the fall of the skull model, however, the MEMS accelerometer loses its capability to estimate orientation since the proof mass has the tendency to move back to the zero-output position inside the micro spring-mass structures [129] in "zero gravity" situations, resulting in unreliable estimation. The gyroscope, in contrast, does not suffer this shortcoming. Though they suffer from white noise and cumulative drift when running for a long period, MEMS gyroscopes offer high-quality measurements of the angular rate that are immune to zero gravity [129, 130]. Performance differences can be seen in the relative data plots from the drop experiments in Figure 4.16. The vertical red dash-dotted lines, the green solid lines, and the red dashed line mark the start point of the fall, the impact time, and the impact end time respectively. These divisions were given by the synchronized triggering signal mentioned in Section 4.2.1.

4.2.3 Sensor Fusion and Data Processing

In Figure 4.16a, the tilt angles which are calculated using data from the accelerometer in the inferior cul-de-sac are plotted in red, while the tilt angles of the reference accelerometer



(a)



(b)

Figure 4.16: The Performance Comparison between MEMS Accelerometer and Gyroscope in Orientation Estimation: (a) Tilt Angles Estimated Based on Accelerometer Data; (b) Tilt Angles Estimated Based on Gyroscope Data.

are plotted in blue. Notice that within the section between the start point of fall and the impact end time, the angles oscillate dramatically and reach unrealistic degree values due to the structural properties of MEMS accelerometers mentioned above.

Yet averaging the angle estimations over the time prior to the start of the model's fall (0 to 0.331 s) and the time after the collision (1.085 s to 1.452 s), we can get the initial and final tilt angles, which are approximately 60.4° and 45.6° for the eye IMU, and 23.3° and 11.2° for the head IMU, respectively for this example. As for the repeated model tests, the average initial and final tilt angles are 60.7° and 46.3° for the eye IMU, and 23.6° and 10.5° for the head IMU, respectively. The standard deviations of the four averaged angles are 0.28°, 0.46°, 0.60°, and 0.26°, respectively. The tilt angles are calculated from the gyroscope data by applying cumulative trapezoidal numerical integration [116] on the output angular rate with initialization at zero degrees. Figure 4.16b displays the calculated tilt angles. Compared with the tilt angles from the accelerometer data, those from the gyroscope data give neat and clear descriptions of the angular positions versus time within the section between the start point of fall and the impact end time. Specifically, the tilt angles experienced a small change in positive direction (the direction away from the impact surface) due to the hand pulling away the brass rod immediately followed by a sharp dropping to the negative, indicating the falling of the skull. Notice that even though both plots share a similar pattern, the tilt angle of the eye IMU oscillates more than that of the head IMU when the impact happened, which corresponds to the passive relative displacement of the eye to the head. Due to lack of absolute angle information, the tilt angles from the gyroscopes are further fused with the tilt angles from the accelerometers.

Figure 4.17 illustrates the data processing procedure to get impact acceleration estimations (see code in Appendix B.5). The raw sensor data obtained from the eye IMU and the head IMU was first translated from digital readings then calibrated by removing the biases. As for the accelerometer data, the biases were calculated by recording and averaging the output of each sensing axis (1500 samples over 1.452 s of period for each axis) when in its stationary zero-output position, and the sum of the outputs of its stationary $\pm 1G$ positions, where $G = 9.80665$. As for the gyroscope, the biases were given by the averaged output of each axis' output when the sensor was still. For both eye and head IMU's, the recording time taken for bias removal equaled the sampling period of the drop test. For the gyroscope, scaling factors were obtained by rotating the sensor 180° around each axis using a servo motor and dividing the gyroscope's rotational angles by 180°. These scaling factors were further used to compensate the tilt angles integrated from the angular rates. The temperature readings given by the integrated temperature sensors of the LSM6DS3 IMU's were also used to calibrate the accelerometers and the gyroscopes since the sensitivities of both were

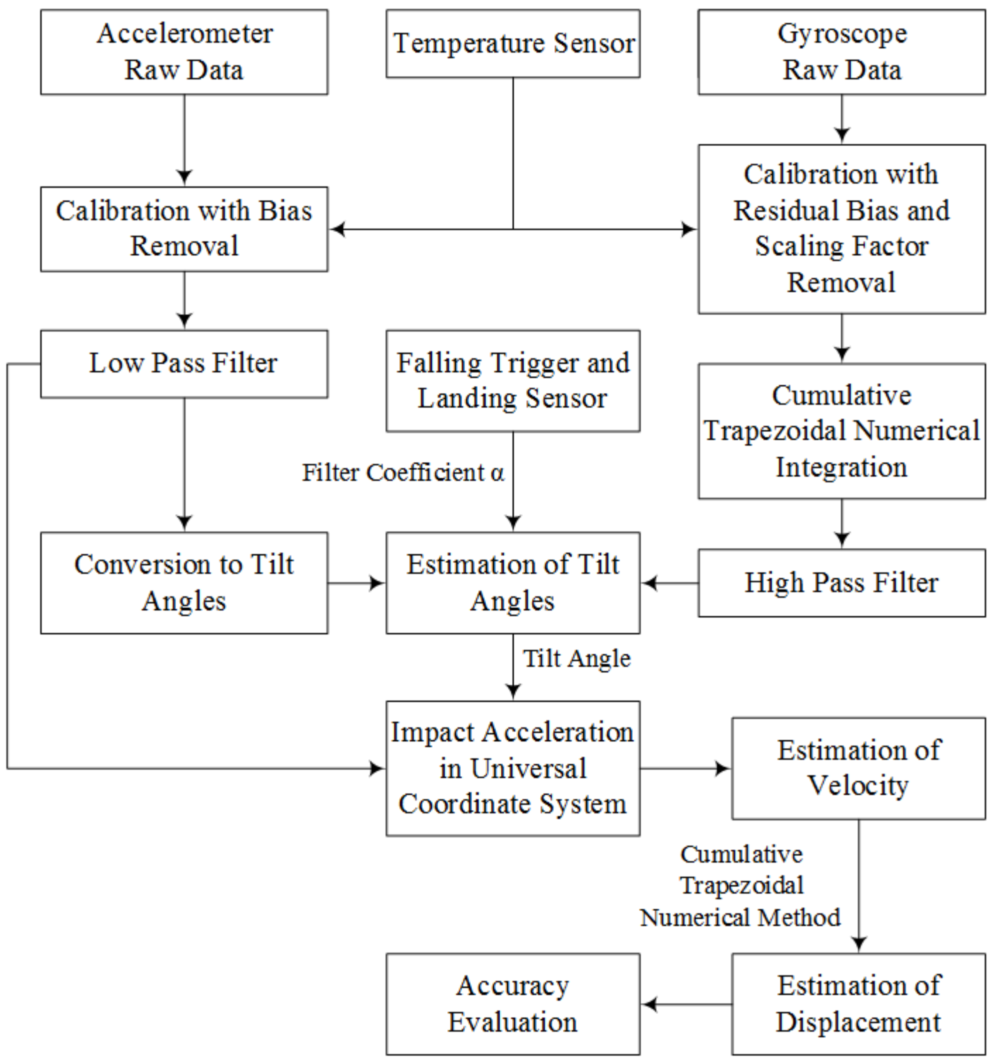


Figure 4.17: A Flowchart of Sensor Fusion for Impact Acceleration Estimation.

affected by the temperature. This would be more useful when applied to the human testing in Section 4.2.4 as the eye IMU and the head IMU were at different temperatures.

The acceleration data was then low-pass filtered to remove noise and substituted into Equation (4.2) to get tilt angles, while the calibrated angular rate data from the gyroscope was integrated using a cumulative trapezoidal numerical method, resulting in tilt angles as well. Data was high-pass filtered afterwards to remove integration bias. A complementary filter was then designed and applied to fuse these tilt angles [131, 132]. The complementary filter can be represented by Equation (4.3),

$$\theta = \alpha\theta_G + (1 - \alpha)\theta_A \tag{4.3}$$

where θ is the tilt angle filtered by the complementary filter, θ_G is the tilt angle calculated from the gyroscope using the cumulative trapezoidal numerical integrator, θ_A is the tilt angle calculated from the accelerometer according to Equation (4.2), and α is the filter constant given by Equation (4.4),

$$\alpha = \frac{\tau}{\tau + T} \tag{4.4}$$

where τ is the time constant and T is the sample period. With the synchronized signals given by the trigger and landing sensor mentioned in Section 4.2.1, τ is set as the time between the beginning of falling and the time when impact ends. The gyroscope data is reliable during this period of time while the accelerometer data becomes more trustworthy after this impact end time. The filter constant α is thus given, as 0.99 in this particular testing to put different weights on the gyroscope data and accelerometer data, which is, in this case, 0.99 on the gyroscope data and 0.01 on the accelerometer data when falling and impact happened. Notice that the two weights of the filter always add to 1 so that the output is an accurate linear estimation. Figure 4.18 shows the results of applying the complementary filter on the data plotted in Figure 4.16. In Figure 4.18, the gyroscope data is plotted in red solid lines, the accelerometer data in blue solid lines and the complementary filtered data in black dashed lines. The time regions are divided in a same manner as that in Figure 4.16.

Notice that the plot of filtered tilt angles not only kept the details of the reasonable angular change when falling and impact happened, but also smoothed out the noisy oscillation during the impact period. This results in the angular estimation being responsive and accurate with respect to either sensor's data alone, and non-sensitive to either the gyroscope drift or the acceleration noises. Improved estimations of the stationary beginning and ending angles are also represented by fusing the two sets of data.

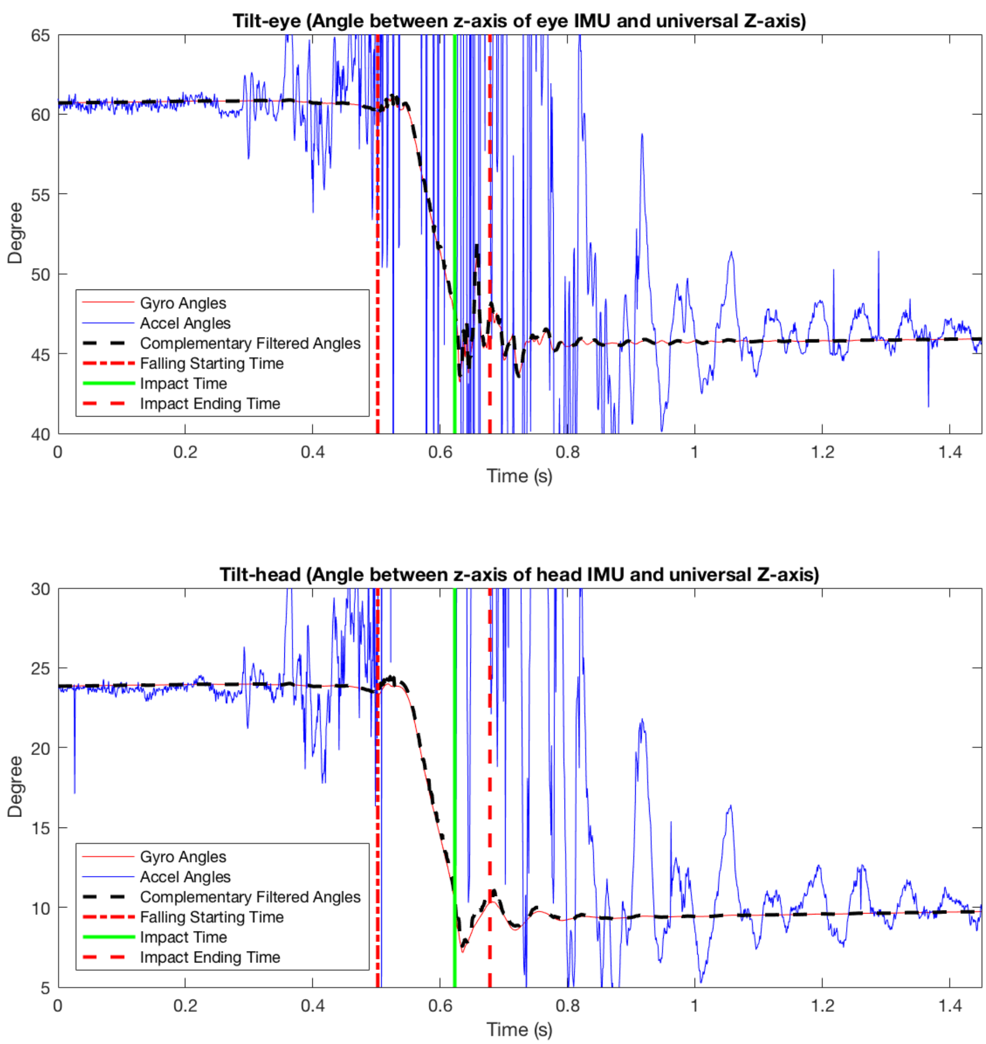


Figure 4.18: Complementary Filtered Tilt Angles in a Skull Model Test.

Since each of the accelerometer outputs can be considered as the projection of the skull's overall acceleration in the universal coordinate system to the certain axis of the accelerometer's local coordinate system, we can fuse known tilt angles with additional acceleration and angular data to make accurate estimations in the universal Z-axis. The overall accelerations of both the eye IMU and the head IMU were thus computed along with their relative accelerations (Figure 4.19). A gravity vector was compensated when calculating the accelerations when falling and bouncing back in the air. As shown in Figure 4.19, a green solid line marks the impact time of 0.623 s and divides the time into the falling section (0.497 s to 0.623 s) and the impact section (0.623 s to 0.678 s). General observation of the accelerations plotted in blue suggests that both IMU's experienced a free-fall like acceleration, a drastic inverse impact acceleration with oscillations, and a relatively converged settling with after-impact ripples. Notice that the IMU in the eye suffered more impact accelerations and longer oscillations compared with the head IMU since the PDMS eye ball was suspended in the gelatin muscle which has lower stiffness than the skull. This corresponds with the fact that the brain, which is fluid, is more injury-prone than the head during a concussion. The ripples at the beginning of the falling section indicate the removal of the brass rod.

Cumulative trapezoidal numerical integrations with 1500 cycles were further applied resulting in the velocities and displacements plotted in red in Figure 4.19. The lowest point of the displacement plots in the impact section, -79.64 mm for the eye IMU and -76.18 mm for the head IMU, correspond with the stationarily measured vertical distances of 71.4 mm and 73.9 mm that the eye IMU and the head IMU have fallen during the experiments, respectively. The differences of the measured falling distances were due to the slightly different positions of the IMU placement on the skull model from the joint of the PVC neck. The relative displacement of the eye to the head at the impact time were calculated by subtracting the lowest-point eye and head displacements, resulting in -3.46 mm. This is also reflected by the larger difference between the estimated vertical displacement of the eye IMU and its measured falling distance than that of the head IMU. Notice that the eye model bounced

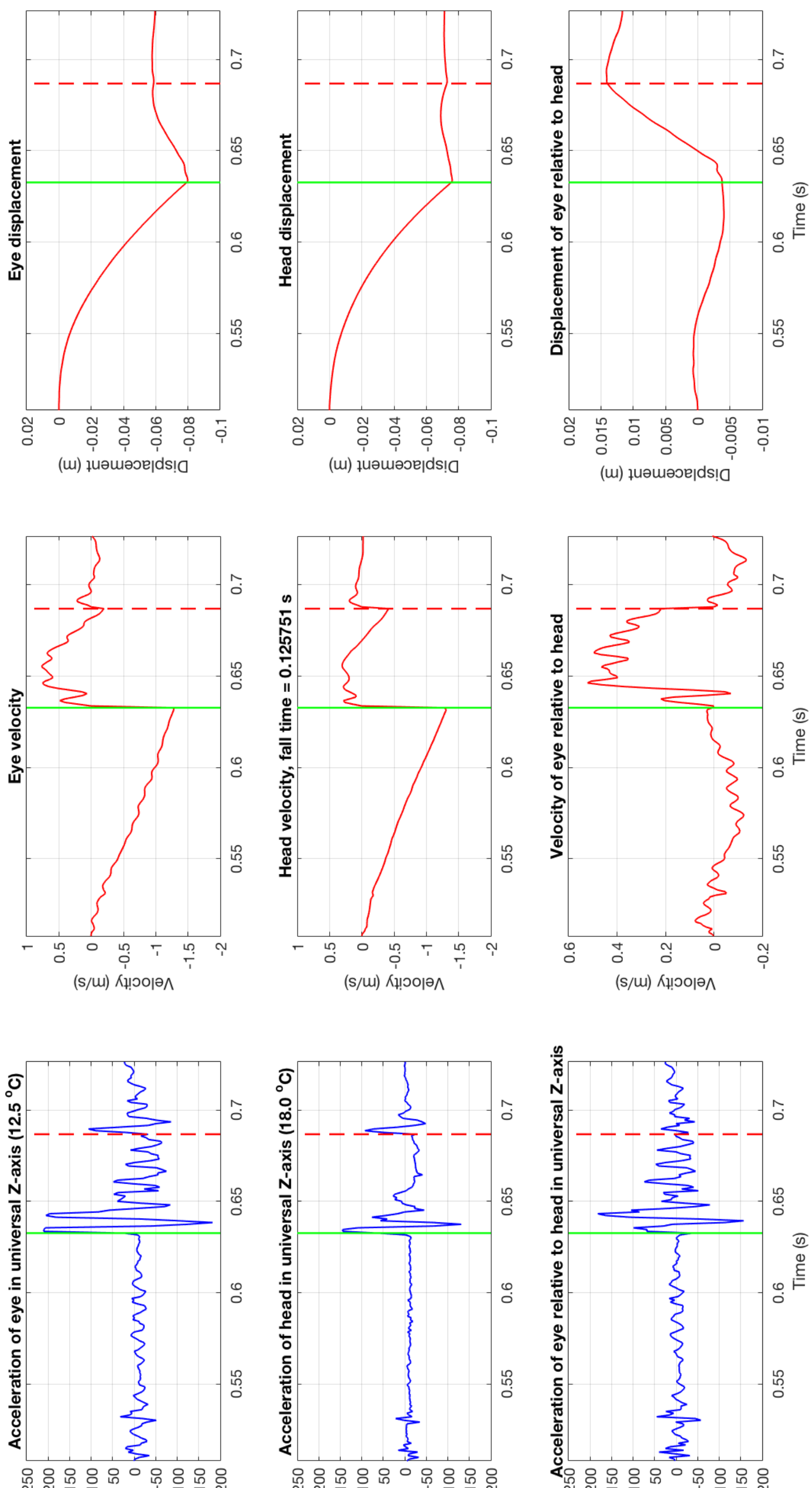
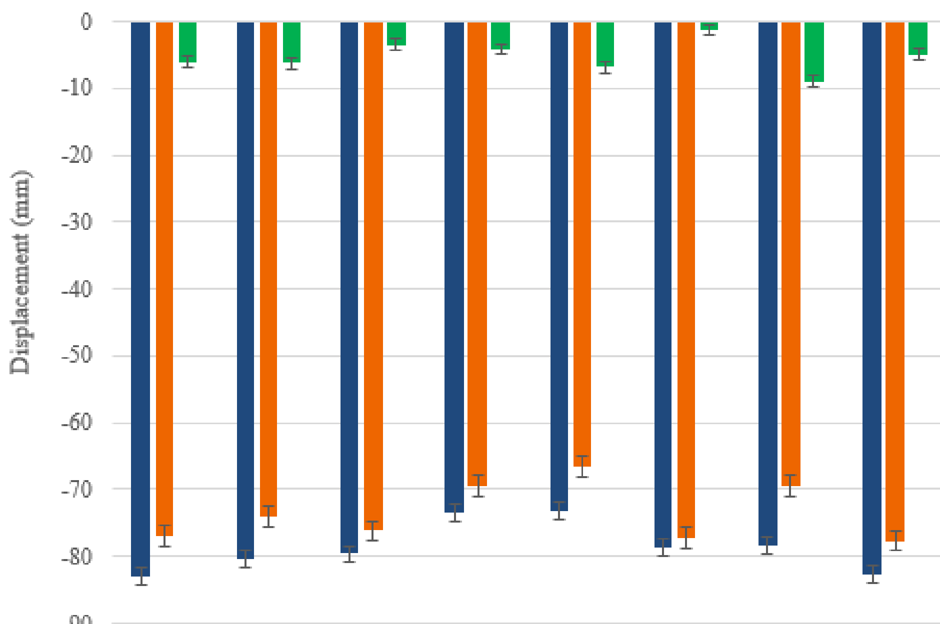


Figure 4.19: Universal Coordinate System Data during Falling and Impact of a Skull Model Test.

up relative to the head after the first impact and dropped down after the second impact, as suggested by the relative displacement plot. This was due to the elasticity of the thin PDMS eyelid which allowed the oscillating eye IMU to move further out of the skull model than into the gelatin socket muscle with the eye ball, before it was damped back to the original position. The above discussed set of results is a typical example of the 8 sets of data collected from the skull model impact experiments which are listed and bar-plotted in Figure 4.20. The relative displacement estimated by our approach was observed to be repeatable. The average relative displacement is -5.26 mm and the standard deviation is 2.34 mm. Since the estimated relative displacement does not have a standard reference, we use the propagated error to study the error (or the uncertainty as named in [133]) of the relative displacement estimation statistically. The estimated eyeball displacement has an uncertainty of 3.64 mm and the estimated skull displacement has an uncertainty of 0.48



Number of Experiment	1	2	3	4	5	6	7	8
Eye Displacement	-82.88	-80.44	-79.64	-73.52	-73.3	-78.61	-78.44	-82.61
Head Displacement	-76.81	-74.14	-76.18	-69.35	-66.46	-77.3	-69.45	-77.66
Relative Displacement	-6.07	-6.3	-3.46	-4.17	-6.84	-1.31	-8.99	-4.95

Figure 4.20: Skull Model Testing Results.

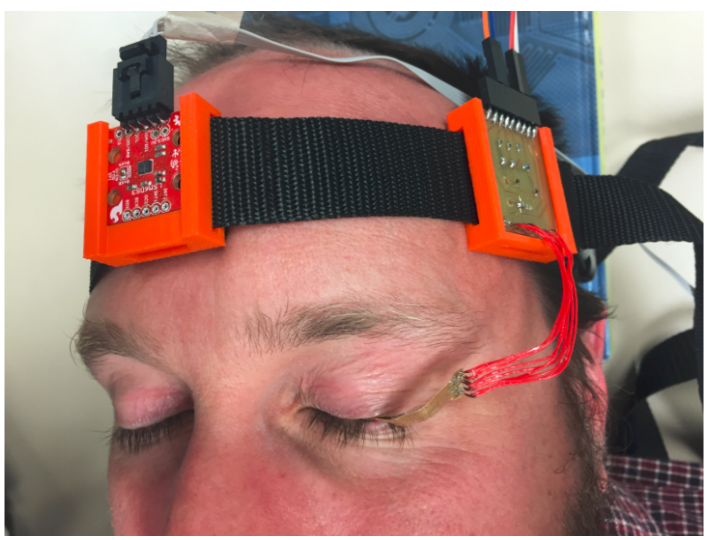
mm. These uncertainty estimates result in a propagated uncertainty of 4.12 mm for the relative displacement estimation.

4.2.4 Tests on Humans

In order to further validate the idea of applying MEMS IMU's to sensing passive eye movement during impact, 2 human volunteers were tested under non-concussive conditions (Figure 4.21). (The study was approved by the Institutional Review Board of UAB and conducted in accordance with the Declaration of Helsinki.) The volunteers laid on their backs on the test bench with a flexible ribbon-style IMU inserted under the eyelid of one eye and the head supported by a hard-cover book with the thickness of 26 mm. The book was pulled out quickly to generate a short free fall of the head. The flexible ribbon-style IMU had been sterilized in PeroxiClear™ for at least 8 hours before the insertion, so it is safe to insert the sensor in the human subject. Similar testing devices and techniques used in the skull model tests were used here for data collection.

Unlike the skull model testing experiments, a start-falling trigger and a landing sensor are difficult to implement on a human head. Fortunately, the gyroscope itself provided a reliable reference to mark these two time points. As shown in Fig. 4.22, the start point of falling (red dash-dotted line) was detected by the concave dimple of the tilt angles plot from the gyroscope (red solid line) which indicated pulling away of the book. In a similar manner, the ending point of the first impact (red dashed line) was detected by the lowest point of the same plot which suggests that the head was bounced back at this moment since a sharp angular change took place. The green solid line marks the impact time according to the accelerometer data which gave the point where a drastic increase of acceleration against the Earth's gravity occurred. Using the same technique of data fusion, the results of this human test was plotted in Fig. 4.23. This time, the eye displacement was read as -33.42 mm, head displacement as -29.11 mm and relative as -4.31 mm.

(a)



(b)

Figure 4.21: Human Test with IMU's.

Instead of a number of oscillations, there was only one dominant spike in the acceleration plots of the human testing results. Especially for the head acceleration, the signal given by the head IMU was damped faster than the signal from the eye IMU. This phenomenon resulted from the presence of human muscles on the head movement, which acted as a low-pass filter of the system. Significant signals, however, were still captured and reasonable estimations of displacement were therefore provided. The relative displacement of the model test tended to be larger than that in the human test not only due to the higher falling height, but also because of the lower stiffness of the gelatin muscles compared with real human muscles. For the same reason, the relative upward bounce displacement of the model testing was also larger than that of the human testing. The similarities turned out to be apparent as well. These include the similar patterns of the accelerations, the velocities and

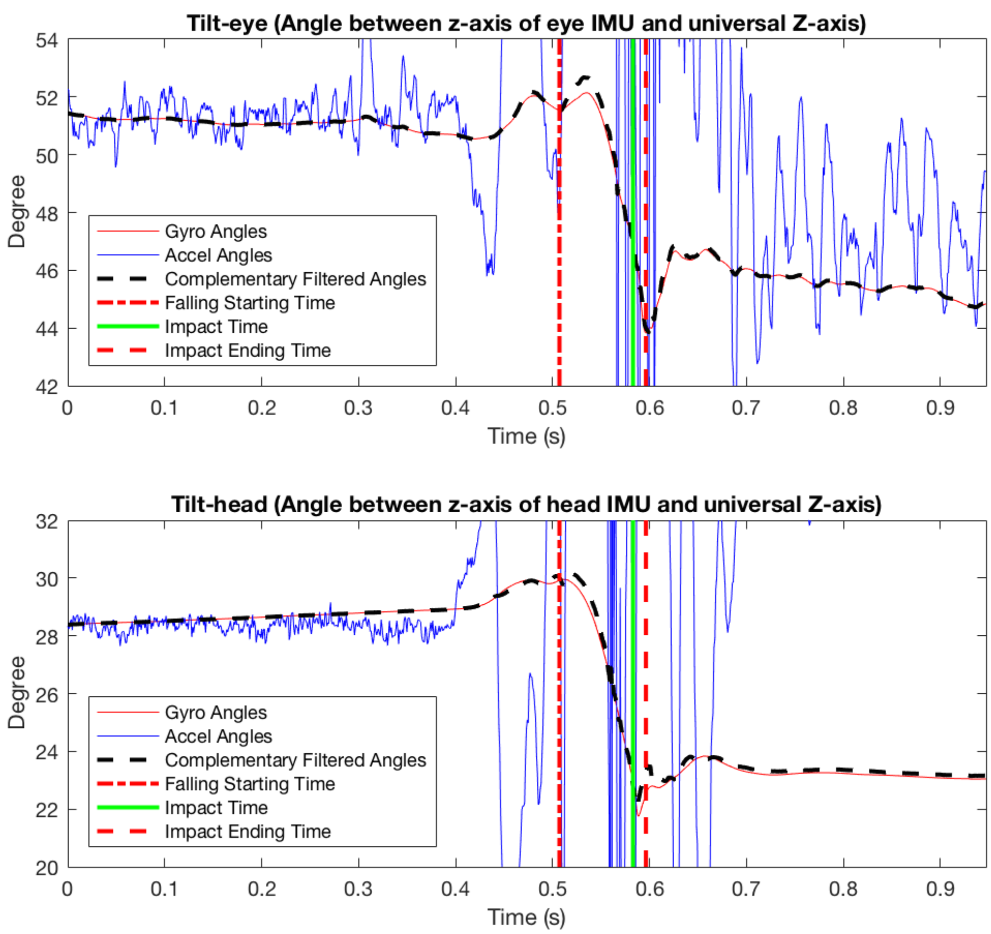


Figure 4.22: Complementary Filtered Tilt Angles in a Human Test.

the displacements, the high accelerations that both the heads and the eyes experienced, the bouncing behaviors when the first impact happened, and the delay of the lowest point of relative displacement to the impact point, indicating the buffering function of the eye related tissues, etc.

After comparing the two types of experiments, we can see that our fabricated skull-brain-eye system acted as a valid model of the human head in the drop-and-impact experiments. The accuracy of our technique was limited by the propagation of uncertainties through the nine steps involved in the indirect measurement of relative displacements [133]. The steps include: (1) the conversion of binary raw thermometer data to centigrade degrees, (2) the updated sensitivities of the accelerometer and the gyroscope by the temperature, (3) the conversion of binary raw accelerometer data to accelerations, (4) the conversion of binary raw angular rate data to angular rates, (5) the calculation of angular rates to angular positions, (6) the data fusion of angular positions and accelerations, (7) the integration from fused accelerations to velocities, (8) the integration from velocities to displacements, (9) the subtraction of the eye and head displacements. Random errors caused by manually pulling the brass trigger rod also affected the results for the model testing in which the relative displacements had a standard deviation of 2.34 mm and a propagated uncertainty of 4.12 mm indicating the precision and the accuracy respectively. Future work will focus on improving the model, isolating as many sources of error in the model system as possible and conducting additional human tests. More volunteers are being recruited at UAB for the next phase of this research, yet it is beyond the scope of this dissertation.

In this section, a technique of sensing passive eye response as a surrogate for brain response to head acceleration was provided by applying MEMS IMU's. A novel design utilizing ribbon-style flexible PCB mounted IMU's was realized for eye insertion to validate the technique. A 3D-printed human skull model along with gelatin brain and muscles, PDMS eye balls and eye lids, and PVC neck model were made to be tested in the head collision experiments. A Teensy MCU with interface circuits was programmed and utilized as a data

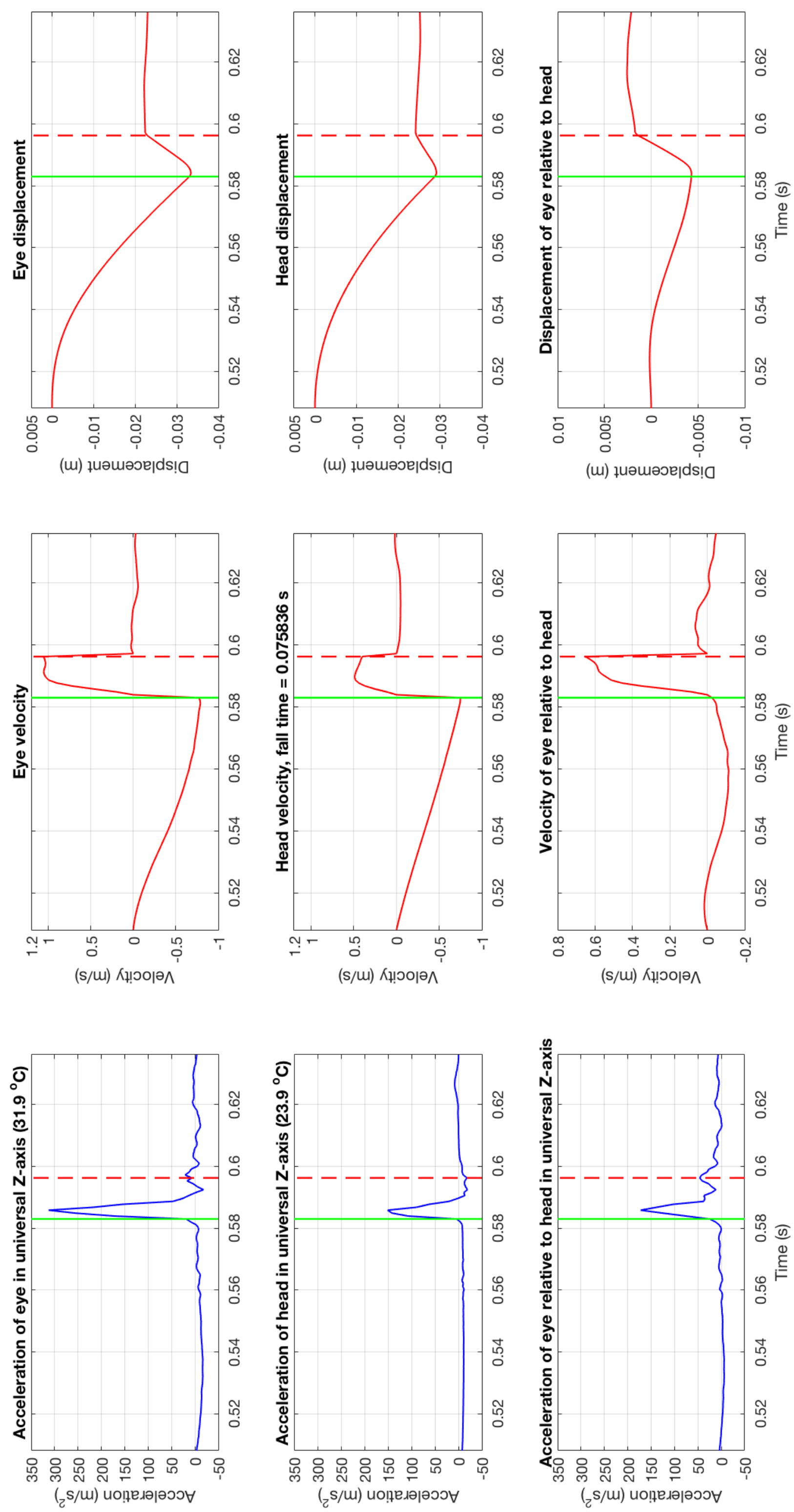


Figure 4.23: Universal Coordinate System Data during Falling and Impact of a Human Test.

acquisition device. Calibrations of the accelerometers and gyroscopes were done and the captured data were further processed by applying cumulative trapezoidal numerical integration, complementary filter, and data fusions. Displacements of the eye, the head, and their relative displacements in the universal coordinate system were estimated and interpreted, which corresponded with the measured values and thus validated the sensing technique. Tests on human volunteers were performed and compared with the model tests, which proved the function of the device. The results of this section were published in [134].

4.3 Semi-Wireless in-Helmet Expansion of the Skull Model Apparatus

As an expansion for the skull-brain-eye model introduced in Section 4.2, a football helmet was placed on the 3D-printed skull with a flexible dipole antenna transmission system embedded on it, which was designed by J. Craig Prather, Michael Bolt and Tyler Horton et al. from the STORM (Sensors, Transducers, Optics, RF and MEMS) Lab of Auburn. The goal for this expansion was to more faithfully emulate real-world situations so as to serve the next phases of this research project in the future.

Most parts of the model test apparatus were identical to the ones in Section 4.2, except for the upgraded flexible ribbon-style eye sensors, a football helmet, a wireless transmitter with a flexible dipole antenna, and a portable drop-and-impact test platform designed for outdoor experiments. Again, two 6-DoF LSM6DS3 IMU's, each containing a 3-DoF accelerometer and a 3-DoF gyroscope, were used to capture data. One of the IMU's was mounted on a flexible ribbon-style PCB made of 1 mil (25.4 μm) thick DuPont^{TM} Pyralux and then coated with silicone conformal coating as seen in Figure 4.24 (layout designs shown in Figure A.4 and A.5). As an improvement, the polyimide substrate material was trimmed into a half-moon shape designed to fit the sensor to the surface of the eyeball and couple the motions of the eye and sensor at the position of the IMU. The sensor's silicone conformal coating created a moisture barrier to protect the circuits and made the sensor less likely to scratch the eye. The second IMU was mounted on a SparkFun^{TM} breakout board as a reference sensor for data calibration and relative displacement computation. The IMU's were attached to the transmitter board with short lengths of wire. The transmitter contained a CR1/3N lithium cell battery, which provided power to the system for approximately 30 minutes. If longer operational life is needed in an environment where replacing the battery is non-trivial, a larger capacity battery could be utilized. The transmitter board also had humidity, temperature, and air pressure sensors which may be utilized in future designs to add more bio-metrics. The transmitter board with antenna can be seen in Figure 4.25. This design allows the antenna to conform with the helmet as shown in Figure 4.26.

For testing, the skull model was inserted into a football helmet containing the transmitter board and flexible antenna and then attached to a PVC emulated spine anchored on a portable platform with leveling feet as shown in Figure 4.27. This setup allowed the skull model to fall in an inverted pendulum orientation, resulting in impact with the platform surface. Repeatable drop-and-impact tests were performed by elevating the skull using

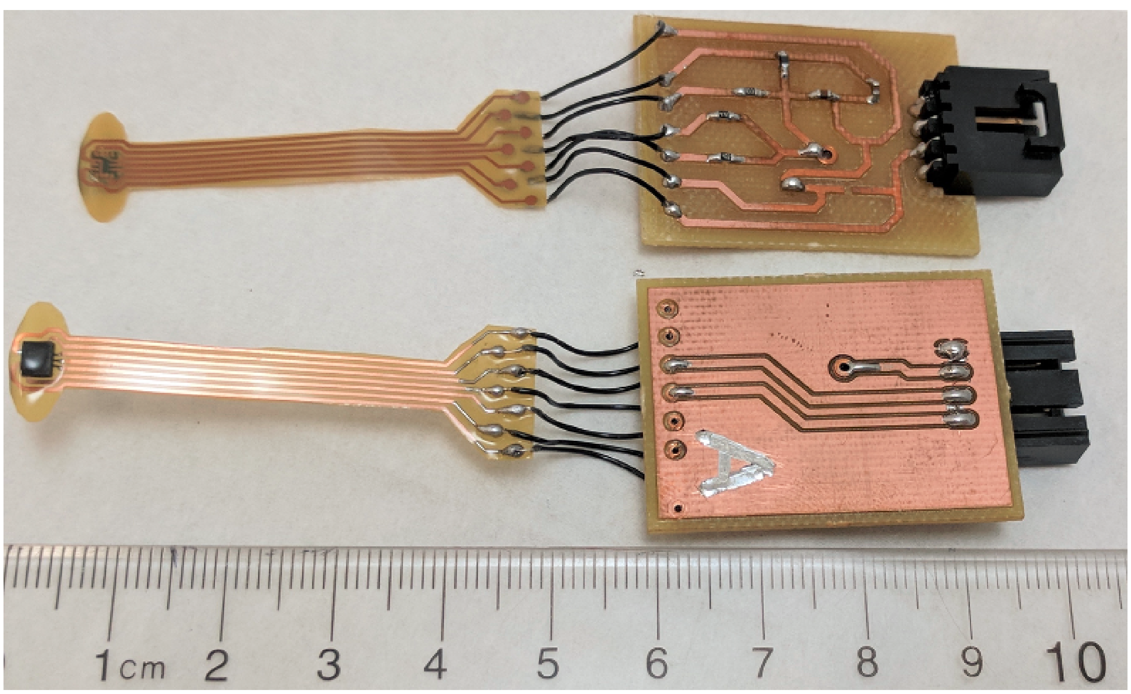


Figure 4.24: Flexible PCB Mounted IMU's with Interface Circuits and Ruler for Scale.

a brass rod and a tiered 3D-printed platform to provide three dropping heights: low (135 mm), medium (155 mm), and high (175 mm). Drops were initiated by removing the brass rod supporting the spine, allowing the skull model and helmet to fall and collide with the platform surface. The brass rod also served as a trigger mechanism to signal the start of the skull's fall, as two electrodes connected to the PVC spine would be shorted while in contact with the brass rod. To signal contact with the platform surface, two electrodes on the back of the football helmet would be shorted on impact in a similar manner using conductive copper tape. This trigger signal was recorded and synchronized with the recorded IMU data.

During testing, the flexible ribbon style sensor was inserted into the eyelid model through the eyelid slot opening and held on the lower surface of the eye replica by the lower eyelid as shown in the inset of Figure 4.27. The reference IMU was held by a 3D-printed PCB holder that was permanently glued to the skull's nose to minimize movement relative to the skull. The x-axes of the two IMU's were adjusted by turning the PVC neck to allow the flexible PCB to align the sensor edges to a reference horizon line on the test platform such that they

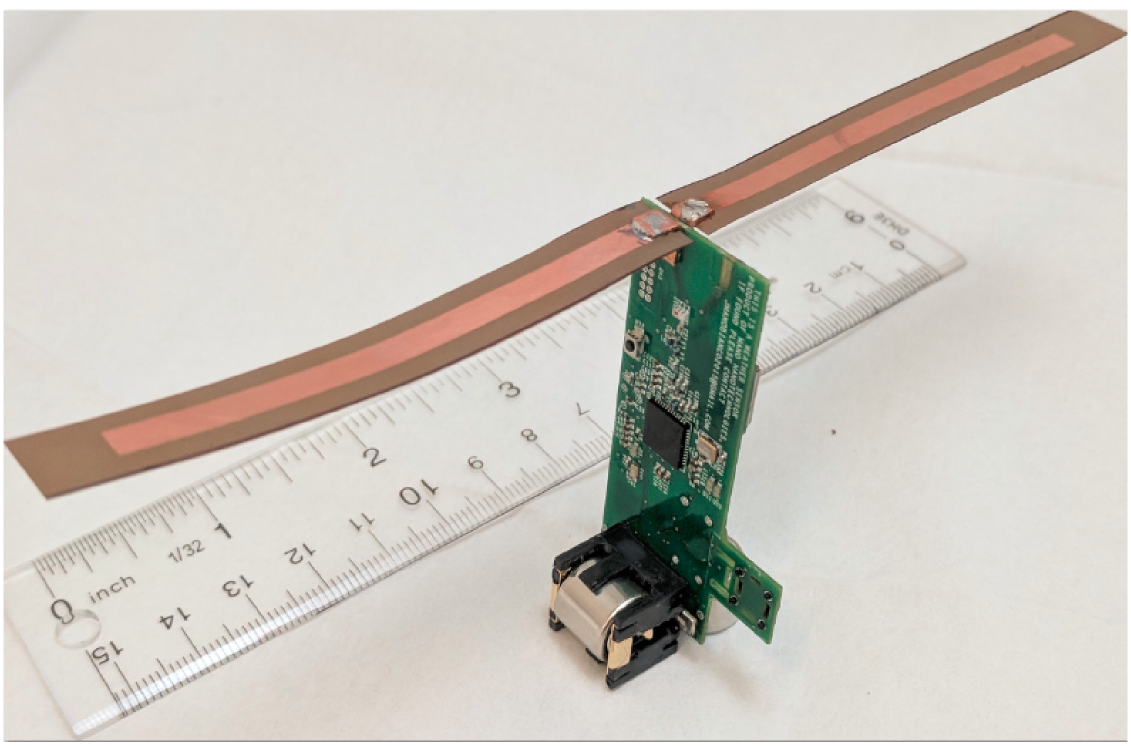


Figure 4.25: Transmitter with Designed Antenna and Ruler for Scale.

lie parallel to each other and perpendicular to gravity. Additional misalignment during drop tests was further compensated for in the data processing algorithm introduced in Section 4.2.

In order to receive live IMU data from the test subject, an MCU with built-in radio frequency (RF) core was utilized on the transmitter board: a Texas Instruments CC430F5137, which contains a Texas Instruments CC1101 sub-GHz ISM band transceiver [135]. The data from the two IMU's attached to the subject are transmitted using 2-frequency-shift-keying

Figure 4.26: Football Helmet with Conformal Flexible Printed Dipole (Top) and Trigger Electrode (Bottom Left).

(FSK) modulation at a Baud rate of 500 kHz to a CC1101-based receiver. The estimated noise floor of the receiver at the current baud rate is approximately -70 dBm. The system transmitted at a power of +10 dBm to stay within the ISM band limitations. This transmission scheme allows for a maximum sampling rate of 499 Hz. The limiting factors of this transmission scheme are the number of IMU's per radio module and the length of wiring required to connect them to the transmitter board. Having more than one IMU per transmitter adds more sample delay, and the length of wire decreases the effective Baud rate of the I^2C bus [136].

The system described above was first tested in the lab. The test set-up of the skull model and sensors can be seen in Figure 4.27. The skull model was dropped from the medium level

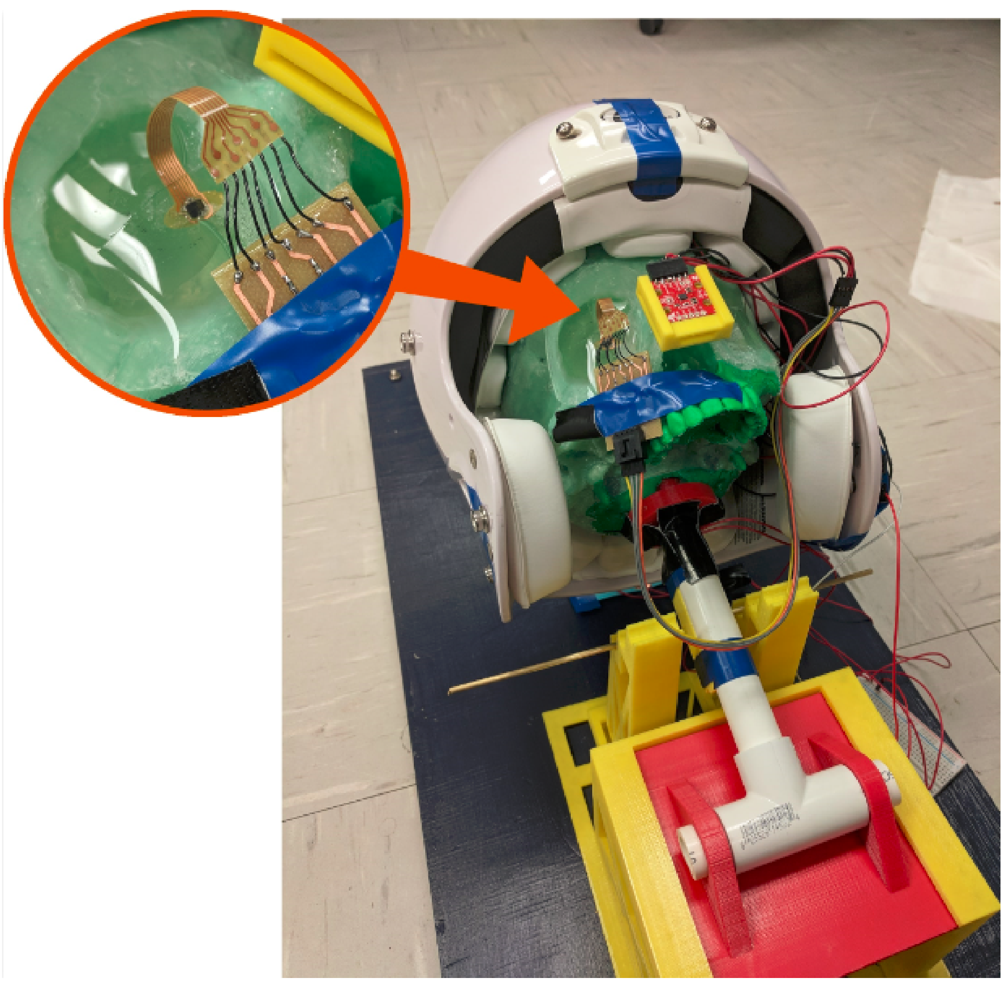


Figure 4.27: Impact Testing System with Inset of Eye Sensor.

(155 mm) and high level (175 mm) 10 times each. Data from the two IMU's and the triggers were transmitted, recorded, and processed using the complementary filter given in Section 4.2, as shown in Figure 4.28. The tilt angles given by the gyroscopes are plotted in red, the tilt angles processed from the accelerometers are plotted in blue, and the complementary filtered tilt angles are plotted in black dashed lines. The vertical red dash-dotted lines, the green solid lines, and the red dashed line mark the start point of the fall, the impact time, and the impact end time respectively; these divisions were given by the synchronized triggering signal. Averaging the tilt angle estimations over the time prior to the start of the model's fall (0 to 0.309 s) and the time after the collision (0.564 s to 0.631 s), the initial and final tilt angles can be found. The values of the tilt angles for the data presented in Figures 4.28 and 4.29 are from the high level drop, which are approximately 69.4° and 28.8° for the eye IMU and 42.7° and 0.6° for the head IMU for the initial and final angles respectively.

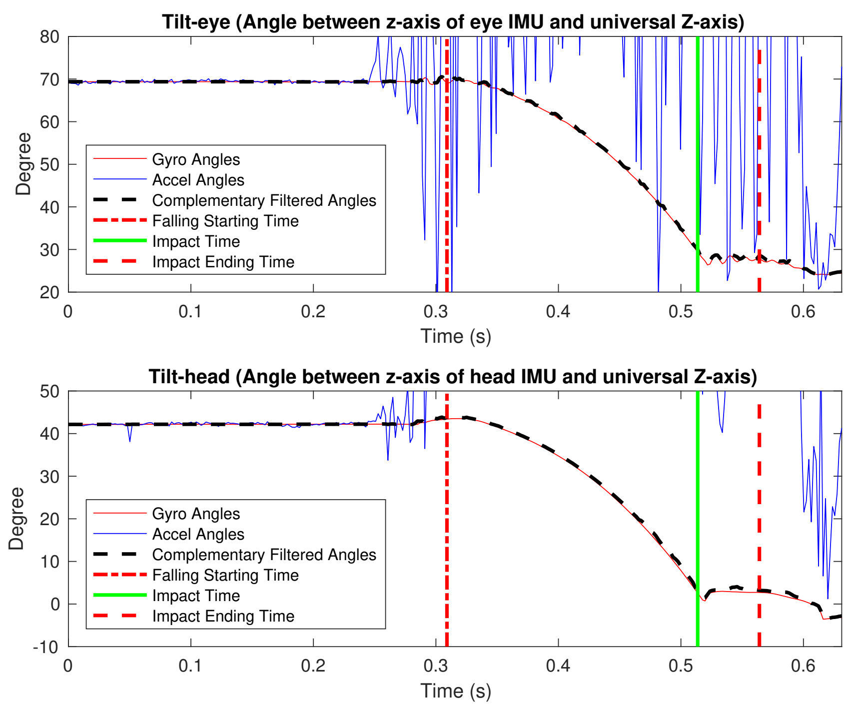


Figure 4.28: Complementary Filtered Tilt Angles in a Skull Analogue Test with a Helmet Mounted RF Transmitter.

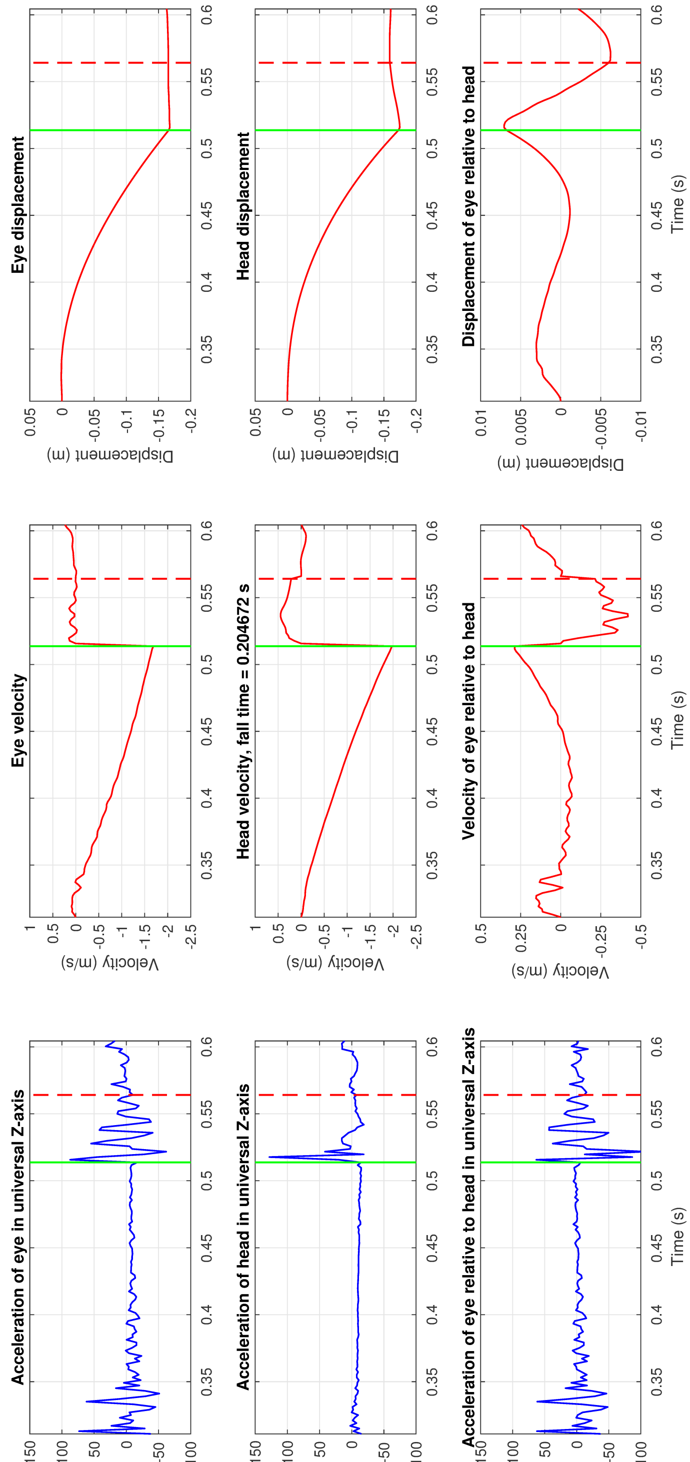


Figure 4.29: Universal Coordinate System Data During Falling and Impact of a Skull Model Test with a Helmet Mounted RF Transmitter.

The tilt angles were then used for estimations of displacement of the eye relative to the head in the direction of the Earth's gravity. Since each of the accelerometer outputs can be considered the projection of the skull's overall acceleration in the universal coordinate system to the axes of the accelerometer's local coordinate system, the known tilt angles can be fused with additional acceleration and angular data to make accurate estimations in the universal Z-axis (the direction of Earth's gravity). The overall accelerations of both the eye IMU and the head IMU were thus computed along with their relative accelerations seen in Figure 4.29. The gravity vector was compensated when calculating the accelerations while falling and bouncing back in the air. As shown in Figure 4.29, a green solid line marks the impact time of 0.514 s and divides the time into the falling section (0.309 s to 0.514 s) and the major impact section (0.514 s to 0.564 s). General observation of the accelerations plotted in blue suggests that both IMU's experienced a free-fall like acceleration, a drastic inverse impact acceleration with oscillations, and a relatively converged settling with after-impact ripples. Notice that the IMU in the eye suffered more impact accelerations and longer oscillations than the head IMU, as the PDMS eyeball was suspended in the gelatin muscle which has lower stiffness than the skull. This corresponds with the fact that the brain, which is suspended in fluid, is more injury-prone than the head during a concussion. The ripples at the beginning of the falling section indicate the pulling away of the brass rod.

Cumulative trapezoidal numerical integrations with 300 cycles were further applied, resulting in the velocities and displacements plotted in red in Figure 4.29. The lowest points of the displacement plots in the impact section, -167.6 mm for the eye IMU and -174.6 mm for the head IMU, correspond with the vertical distances of 164.1 mm and 175.8 mm that the eye IMU and the head IMU fell during the experiments respectively. The relative displacement of the eye to the head at the impact time was calculated by subtracting the lowest-point eye and head displacements, resulting in -6.2 mm at 0.570 s. Note that the eye model oscillated relative to the head before the first major impact, sunk into the socket after the first impact, and started to bounce back when the first impact was finished, as suggested by the relative displacement plot. This happened before the eye replica was damped back to its original position, due to the elasticity of the gelatin socket muscle. The above discussed set of results is a typical example of the 10 repeatable sets of data collected from the experiments in which the skull model was dropped from the high level.

The relative accelerations and displacements of the eye to the head suggests that sensing the passive eye motion in response to the head trauma caused by impacts is a potential method for in-field concussion monitoring and diagnosis. Furthermore, this flexible antenna and transmission hardware allowed the system to function in a wireless and wearable manner, enabling this sensing technique to be utilized for in-field applications.

Chapter 5

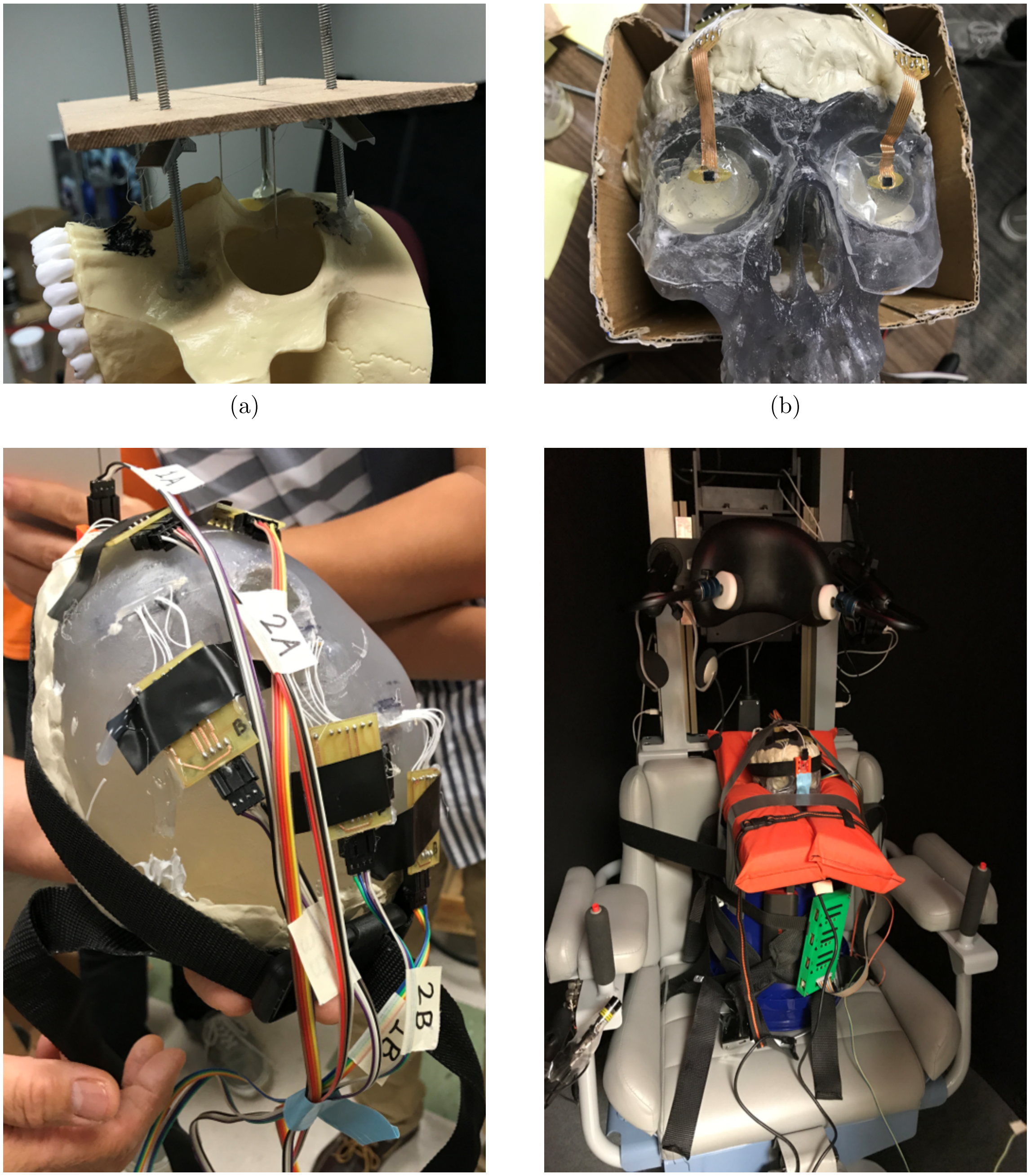
Sensing Brain Response to Head Rotation

More experimentation had been completed to expand the idea of sensing passive eye response as a surrogate for brain response to head acceleration from linear acceleration to angular/rotational. Another step forward was to capture the accelerations of the brain using inserted IMU's. This required a more realistic phantom brain instead of the one in Figure 4.10b, and limited the experiments to be done on models instead of human volunteers.

5.1 Experiments with VOR Chair

The need for a more realistic gelatin phantom brain made the procedure of preparation for experiments different from what has been previously described. The 3D-printed plastic skull had multiple openings that needed to be closed to create the gelatin brain. Oatey™ plumber's putty and tube and tile 100% silicone were used to prepare the skull model for creation of the phantom brain and eyes by filling these holes. Furthermore, the eye sockets were also filled with putty to prevent leaking during the creation of the gelatin eyes. Once all the holes were filled, Dremel™ and drill power tools were used along with multiple screws to drill four small holes into the skull around the eye sockets to secure the phantom eyeballs in place (Figure 5.1a). Finally, three slits and one hole were created in the top piece of the skull using a Dremel™. The three slits were where the IMU sensors were inserted and the one hole is for pouring of the gelatin. The three slits for the IMU's were placed on a diagonal.

1500 mL of 5% gelatin type A was created by putting 75 g of gelatin in 750 mL of water, boiling this mixture, and then adding 750 mL of room temperature water to the solution immediately after boiling. The gelatin was allowed to settle for a few minutes to allow it to cool and to allow for the bubbles to subside within the gelatin. This gelatin mixture was



(a) (b) (c) (d)

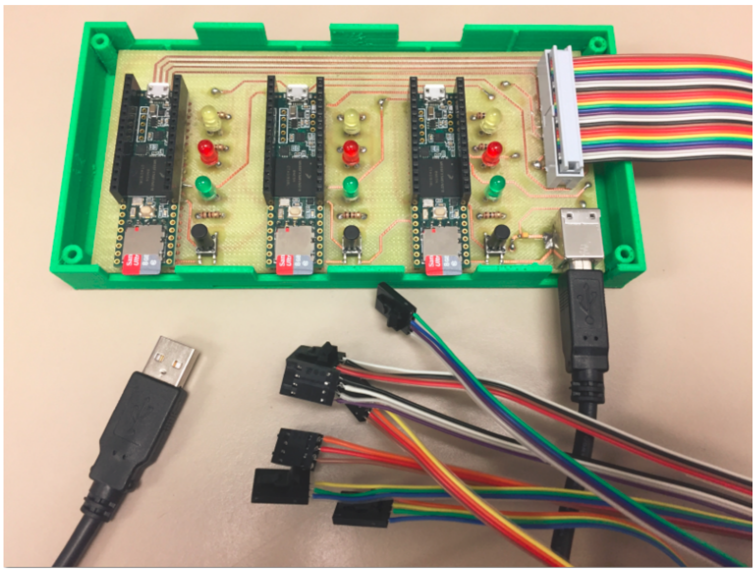
Figure 5.1: The VOR Chair Test Apparatus: (a) Placements of PDMS Eyeballs with an Alignment Device; (b) Placements of Ribbon-Style Flexible PCB Mounted IMU's on the Eyeballs; (c) Placements and Connections of Ribbon-Style Flexible PCB Mounted IMU's in the Gelatin Brain; (d) Placements of the Skull Model on the VOR Chair with All IMU's Connected and Synced to a DAQ.

poured into the skull through the pour hole and then the skull was placed in a 4°C fridge overnight to allow for it to solidify.

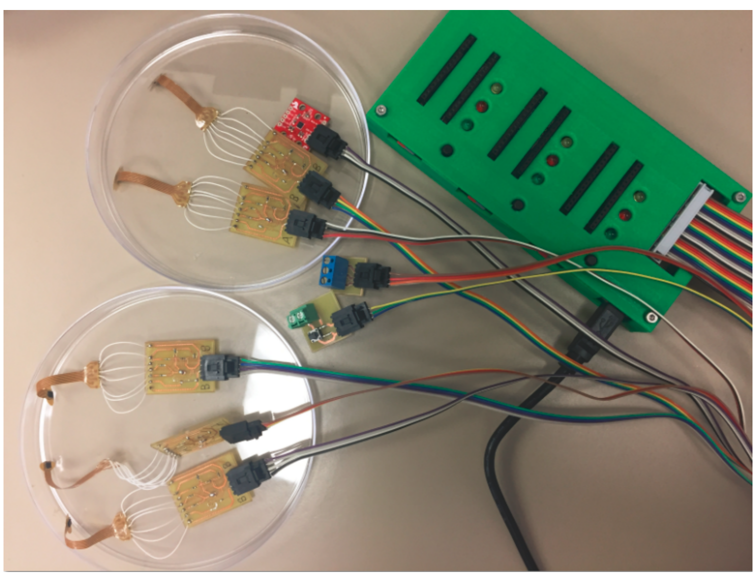
The next day, once the gelatin in the brain had totally solidified, the phantom eyes were created. The eyeball and eyelid replicas were made in the same way as mentioned in Section 4.2. In preparing the phantom eyes, a suspension platform was mechanically attached on the 3D-printed phantom skull using the four small holes mentioned above, and held the two phantom eyeballs in the correct anatomical position while the gelatin inside the sockets were solidified (Figure 5.1a). Both 2% and 4% gelatin solutions were used for the two different eyeballs. The same procedure was used to make this gelatin as the gelatin for the brain, which was done with 50 mL instead of 1500 mL. This gelatin was put in the eye sockets and allowed to cool in a fridge at 4°C for 1 hour to solidify. A piece of PDMS eyelid was then glued to the skull around each eye socket, had a slit cut in the center, and had an IMU placed within the slit where it was touching both the eyelid and the eyeball.

Six of the flexible ribbon-style PCB mounted MEMS IMU's (Figure 4.24), which had been used in the translational collision experiments, were used in the head rotation tests. Among the six IMU's, one was tightly fastened on the forehead as the reference which moves with the skull, two were inserted through the horizontal slits of the eyelids and held tightly on the eyeballs by the eyelids (Figure 5.1b), and the other three were inserted in the gelatin brain in the skull cavity through three slits on the skull top (Figure 5.1c). The positions of these three slits were on a diagonal so that there was one IMU in each of the three typical locations of the brain. There was one slit in the middle of the brain, one in the front left of the brain, and one in the back right of the brain. The IMU sensors were placed in the top of the skull and were manipulated to ensure they all faced the same direction and were within the same sagittal plane. They were then taped down using electrical tape, and then the top of the skull is stuck on to the bottom using silicon and plumber's putty.

A DAQ containing three Teensy™ 3.6 MCUs was created to control and communicate with the six IMU's via I^2C buses, acquire sensing data, and log the data in a micro SD card for further processing (Figure 5.2). The MCUs were plugged on a motherboard (Figure A.6) and synchronized by sharing the same trigger interrupt. Each MCU hosts two IMU's. Through the MCUs, the gyroscopes were programmed to operate with a sensing range of ±2000 deg/s and a sensitivity of 70 mdps/LSB. The sampling rates and bandwidths of the gyroscopes were set as 1.66 kHz and 400 Hz respectively to obtain the best performance



(a)



(b)

Figure 5.2: Photographs of Customized Teensy™ 3.6 DAQ in a 3D-Printed Box Cabled to Six IMU's, a Trigger Button, and a TTL Output.

of the IMU's. The ODR was set as 6664 Hz. The bandwidth is determined by the ODR selection suggested by the data sheet to set an anti-aliasing filter for high performance. All the sensing data, including temperature readings from the six IMU's, were buffered in the 256 KB RAM of each Teensy 3.6 MCU before being logged to its micro SD card. The overall sampling rate was 1 kHz which was sufficient for this application.

A VOR chair, owned by the Vision Sciences Department at the UAB, was used to actuate rotational motions of the skull phantom, so as to collect data about the movement of the IMU's within the brain and on the eyes (Figure 5.1d). Two different data collection tasks were performed with this chair. The first task was a simple 60^o sinusoidal movement that ranged from 0.32 Hz to 1.75 Hz. The second task was a more complicated "hit" motion that consists of a very quick jerk in a random direction, either clockwise or counterclockwise. The peak velocity during the hit-motion task was 160 ^{o}/s and the peak acceleration was 1025 ^{o}/s^2.

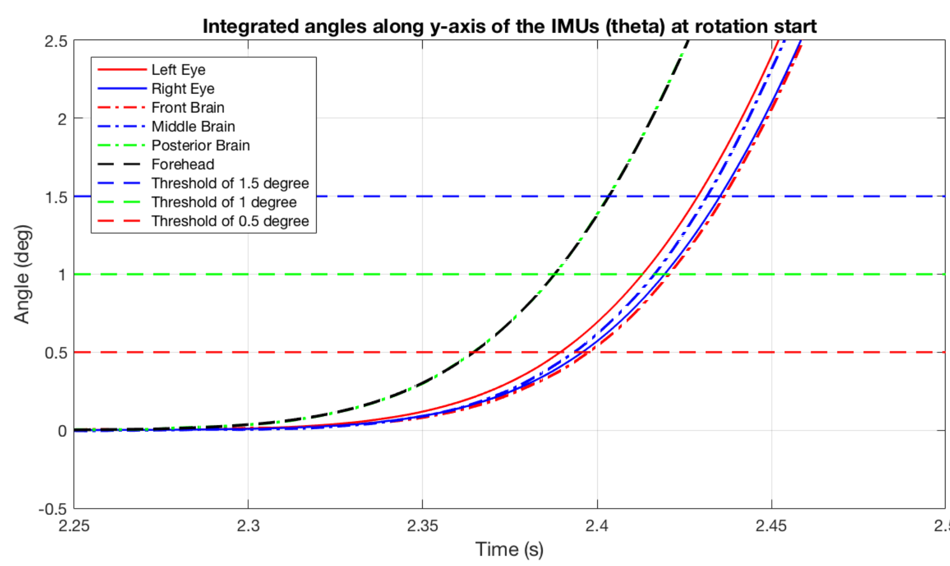


Figure 5.3: A Plot of Rotation Angles around the y-Axes of the IMU's against Time in a VOR Chair Hit-Motion Experiment at Rotation Start.

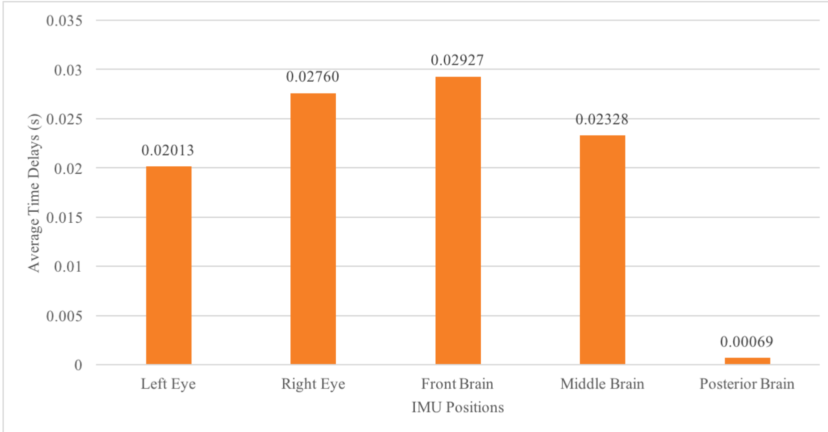
5.2 Data Processing and Analysis of Hit-Motion Delay

The hit-motion experiments provided a better understanding of the time delays associated with the start of rotation of the test IMU's compared to the start time point of the reference IMU. Angular accelerations recorded by the gyroscopes were integrated to get time-varying rotation angles. Figure 5.3 shows a plot of rotational data around the y-axes (angles of θ) collected by all the six IMU's during a typical counterclockwise hit-motion at the start. The y-axes were studied because the IMU's were deployed in a way that the majority of rotations happened around the y-axes. Three thresholds of 0.5°, 1°, and 1.5° were applied to study the time delays. Though varying in different rotational thresholds, the time delays compared to the reference forehead IMU, from short to long, all tend to follow the order of posterior brain, left eye, middle brain, right eye, and front brain, as shown in Figure 5.4. This tendency was repeatable in the ten hit-motions in the experiment of four clockwise rotations and six counterclockwise rotations. The statistics of all the processed data with their averages and standard deviations (STD) are recorded in Table 5.1 (see code in Appendix B.4).

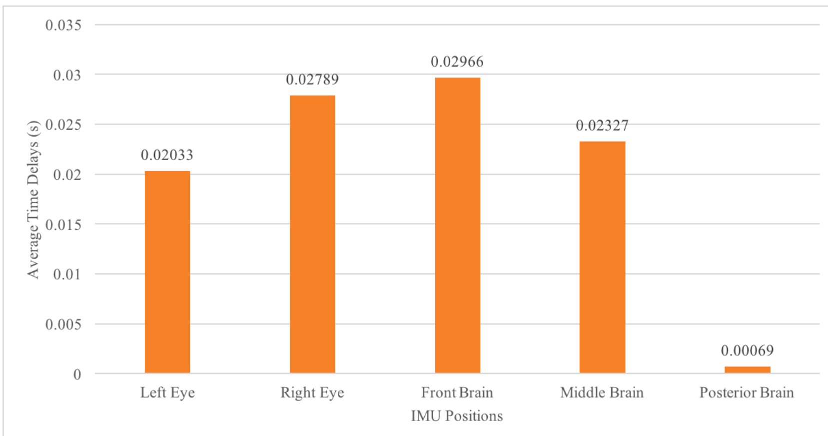
The results proved that the rotational motions of both eyes were good reflections of the rotations of the brain since the time delays of both eyes from the skull rotation fell into the range of the time delays of the different parts of the brain. Specifically, in average, both eye delays were longer than the posterior brain delay, and shorter than the front brain delay. The posterior brain having the minimum delay while the front having the maximum delay correspond with the fact that the posterior brain IMU was closest to the inner wall of the skull back and the rotation axis of the VOR chair, therefore the motion (see Figure 5.1d) coupled better to the skull, which was fastened on and moved along the chair. Although the IMU's on the eyeballs were further away from the rotation axis than the front brain, the confinement of the sockets, which have much smaller cavities than the skull, acted in a way to accelerate the eyeball rotations to match the pace of the brain. One should also notice that the left eye had shorter delay than the right eye due to the greater stiffness of the 4% gelatin than the 2% gelatin.

Threshold	Number and Type	Left Eye	Right Eye	Front Brain	Middle Brain	Posterior Brain
0.5	1 CCW	0.023523	0.027444	0.028424	0.027444	0.000980
	2 CW	0.025491	0.029412	0.032354	0.026471	0.000980
	3 CCW	0.006871	0.006871	0.007852	0.009815	0.000982
	4 CCW	0.024528	0.030414	0.032377	0.028452	0.000981
	5 CW	0.003925	0.006869	0.008831	0.006869	0.000981
	6 CCW	0.038350	0.056050	0.058017	0.042283	0.000000
	7 CW	0.038447	0.051263	0.053235	0.042391	0.000000
	8 CCW	0.005895	0.006877	0.007860	0.008842	0.000982
	9 CCW	0.018599	0.034261	0.035240	0.022514	0.000000
	10 CW	0.015707	0.026505	0.028468	0.017670	0.000982
	Average	0.02013	0.02760	0.02927	0.02328	0.00069
	STD	0.01243	0.01737	0.01761	0.01279	0.00047
1.0	1 CCW	0.024503	0.028424	0.029404	0.027444	0.000980
	2 CW	0.024510	0.029412	0.031373	0.027452	0.000980
	3 CCW	0.007852	0.007852	0.008834	0.009815	0.000000
	4 CCW	0.025509	0.031395	0.033358	0.028452	0.000000
	5 CW	0.003925	0.006869	0.008831	0.006869	0.000981
	6 CCW	0.038350	0.056050	0.058017	0.041300	0.000983
	7 CW	0.039433	0.052249	0.054221	0.042391	0.000986
	8 CCW	0.005895	0.006877	0.008842	0.008842	0.000982
	9 CCW	0.018599	0.034261	0.035240	0.022514	0.000000
	10 CW	0.014725	0.025523	0.028468	0.017670	0.000982
	Average	0.02033	0.02789	0.02966	0.02327	0.00069
	STD	0.01256	0.01742	0.01750	0.01266	0.00047
1.5	1 CCW	0.024503	0.028424	0.029404	0.026464	0.000980
	2 CW	0.026471	0.030393	0.032354	0.027452	0.000980
	3 CCW	0.007852	0.007852	0.008834	0.009815	0.000000
	4 CCW	0.025509	0.032377	0.033358	0.028452	0.000000
	5 CW	0.003925	0.007850	0.008831	0.006869	0.000000
	6 CCW	0.038350	0.057033	0.058017	0.041300	0.000000
	7 CW	0.039433	0.053235	0.054221	0.042391	0.000000
	8 CCW	0.005895	0.007860	0.008842	0.008842	0.000000
	9 CCW	0.019578	0.034261	0.035240	0.021535	0.000979
	10 CW	0.015707	0.026505	0.028468	0.017670	0.000982
	Average	0.02072	0.02858	0.02976	0.02308	0.00039
	STD	0.01258	0.01751	0.01752	0.01264	0.00051

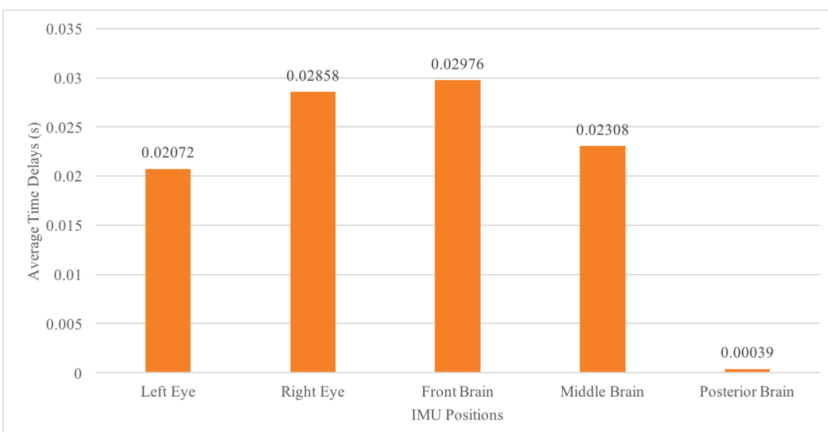
Table 5.1: Time Delays of Rotation Start Points of Eye and Brain IMU's Relative to the Reference IMU on Forehead under Three Thresholds during VOR Chair Hit-Motion Tests (All Units in s).



(a)



(b)



(c)

Figure 5.4: Average Time Delays of Rotation Start Points Compared to the Reference IMU on Forehead with a Threshold of : (a) 0.5 s; (b) 1 s; (c) 1.5 s.

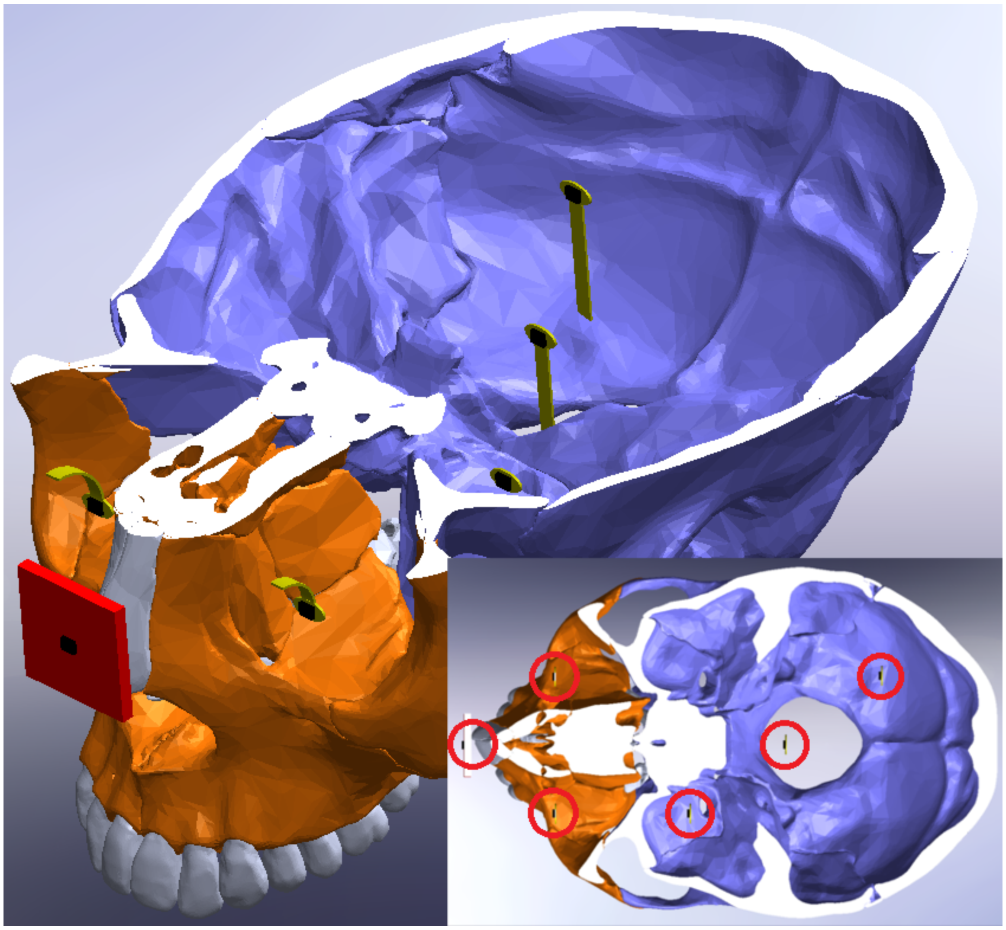
Chapter 6

Correlating Passive Eye and Brain Motions

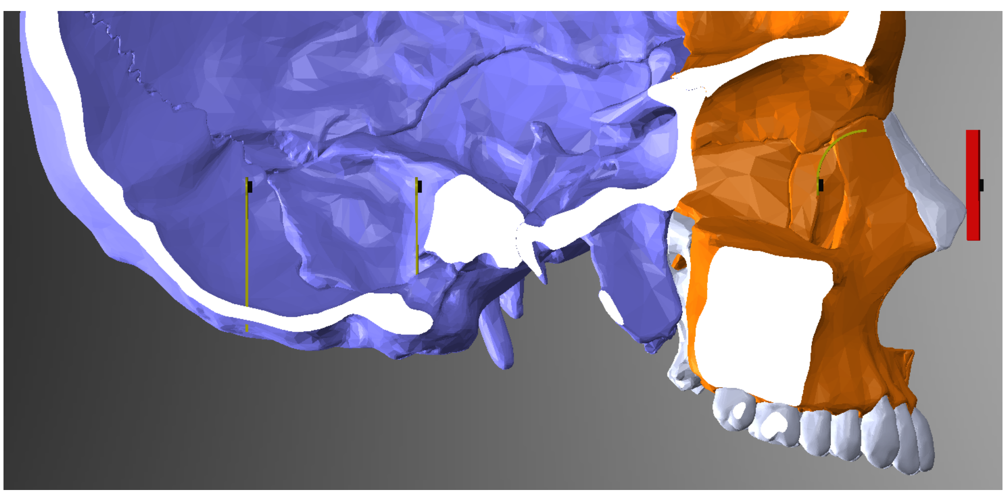
Since the PER has been proven to have relative linear and angular accelerations against the head in high impact environments where concussions may occur, further experiments were done on the fabricated skull-brain-eye system to find the correlation between the PER and brain motions under impact. In this chapter, the DAQ and sensing system used in Chapter 5 was further improved in accuracy. The system was applied in the drop-and-impact test apparatus designed in Section 4.3 except that IMU's were placed in a gelatin phantom brain. Regression models were built to study the correlations between the PER and brain motion.

6.1 Experiment Apparatus

Although the sensor fusion technique corrects the misalignments of IMU axes as introduced in Section 4.2.3, improved experimental hardware and a refined procedure are beneficial to obtain more accurate raw data which are also easier to process. In the experimental apparatus of this chapter, the LSM6DS3 IMU's were used, one on a SparkFun™ breakout PCB and five mounted on the flexible ribbon-style sensor PCB as shown in Figure 4.24. As illustrated in Figure 6.1, the reference IMU was attached on the nose of the skull model, two flexible IMU's were attached on the right eye (RE) and left eye (LE) replicas, and three flexible IMU's were inserted in the phantom brain at three typical positions on a diagonal: front brain (FB), middle brain (MB), and posterior brain (PB). All the IMU's were aligned with their corresponding sensing axes parallel to one another, and with their centers on the same horizontal sagittal plane (i.e. x-z plane referring to Figure 4.15), resulting in their z-axes in a direction pointing to the front of and vertical to the face of the skull, x-axes



(a)



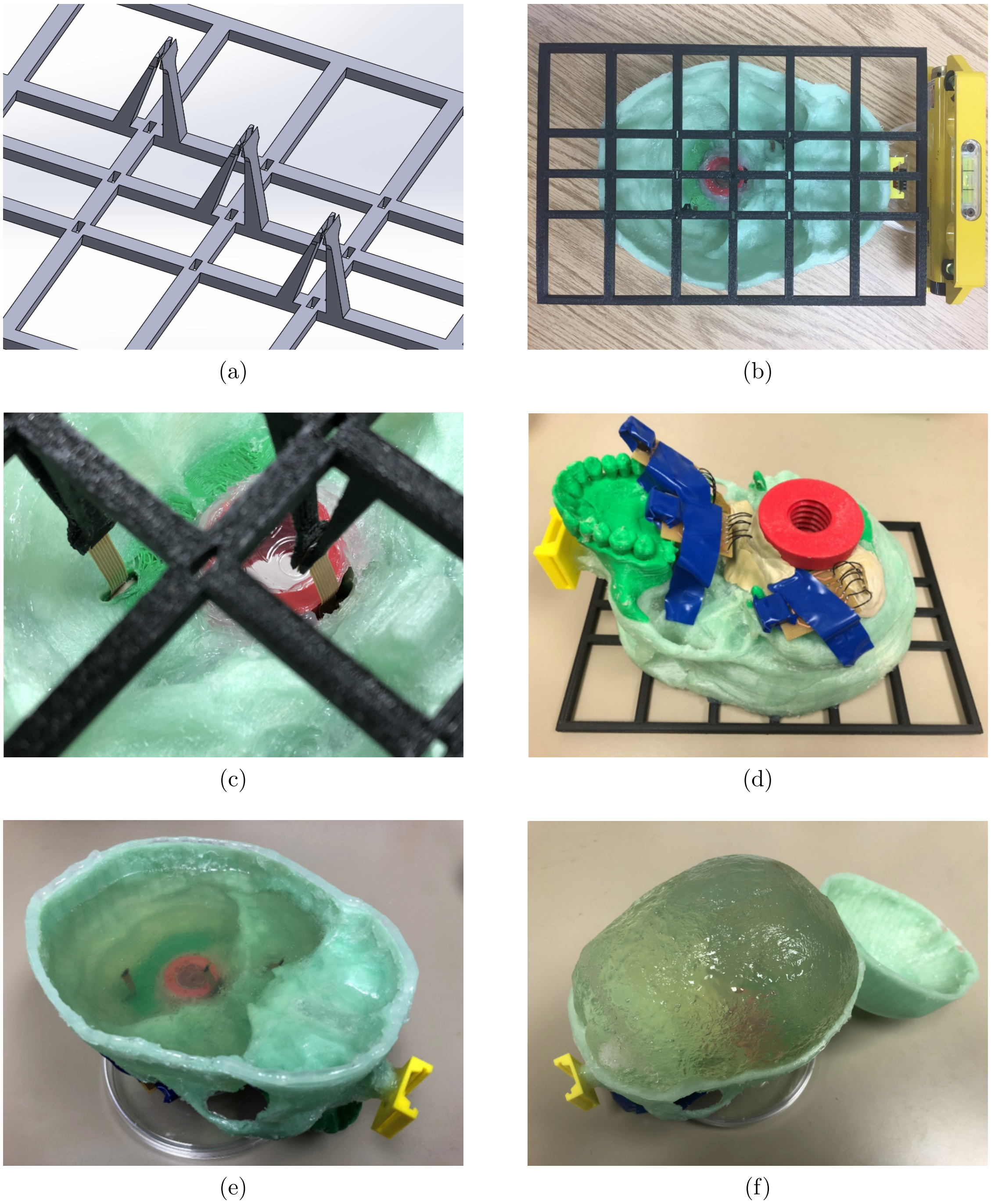
(b)

Figure 6.1: Illustration of IMU Positions.

pointing to the left side of the skull, and y-axes upward. Red circles in the cross-section view of Figure 6.1a highlight relative positions of the six IMU's on the x-z plane. Notice that the IMU's at FB, MB and PB were evenly separated on a diagonal, also the z-axes of IMU's at RE and PB, LE and FB, reference and MB, overlapped in x-axes respectively.

The alignment of the brain IMU's was achieved by a 3D-printed aligner frame designed in SolidWorks™ as shown in Figure 6.2a. Three pairs of clamps were designed to fit the shape and dimensions of the flexible ribbon-style PCB mounted IMU's, and to hold the three brain IMU's in the desired positions both horizontally and vertically. Observation windows were opened on the aligner frame to calibrate the alignments. Silicone glue was used to attach the frame onto the lower half of the skull model to ease later removal, while super glue was used to attach the reference IMU holder so as to firmly couple its motion to the skull (Figure 6.2b). A level and a laser pointer were used to align the frame with the reference IMU holder before the glues cured. Brain IMU's were inserted to the lower half of the skull through slits opened under the desired positions, and then held by the clamps (Figure 6.2c). The slits were then sealed with plumber's putty for water proofing (Figure 6.2d). It took three steps to make the gelatin phantom brain afterwards. Firstly, gelatin solution of 4% mass ratio was poured into the lower half of the skull model just covering the IMU's to form the first layer. Secondly, when the first layer was solidified, the silicone glue on the aligner frame was cut through by a razor blade so that the frame could be removed carefully without shifting the IMU's. A thin second layer of gelatin was then formed to fill the small holes left on the first layer (Figure 6.2e). Thirdly, the two halves of the skull model was glued together by silicone glue, a hole was drilled at the forehead, and the same gelatin solution was poured into the cavity to fill up the skull with the third layer. The formed brain phantom after tests is shown in Figure 6.2f with the skull opened.

The finished test apparatus is shown in Figure 6.3. Most of the settings were identical to the ones inroduced in Section 4.3. The reference IMU was inserted in the holder which was fixed on the skull model. The two eye IMU's were placed on the PDMS eye replicas, held



(a) (b) (c) (d) (e) (f)

Figure 6.2: Alignment of IMU's: (a) The 3D CAD Model of the IMU Aligner Frame Made in SolidWorks^{TM}; (b) A 3D-Printed Aligner Frame (Black) Attached to the Skull Model (Green) while Aligned to the Reference IMU Case (Yellow); (c) IMU's Held in Position by the Aligning Frame; (d) Sealing the Slits where Ribbon-Style Flexible PCB Mounted IMU's Inserted Through; (e) The Second Layer of Gelatin Formed; (f) The Gelatin Phantom Brain (Skull Opened after Tests).

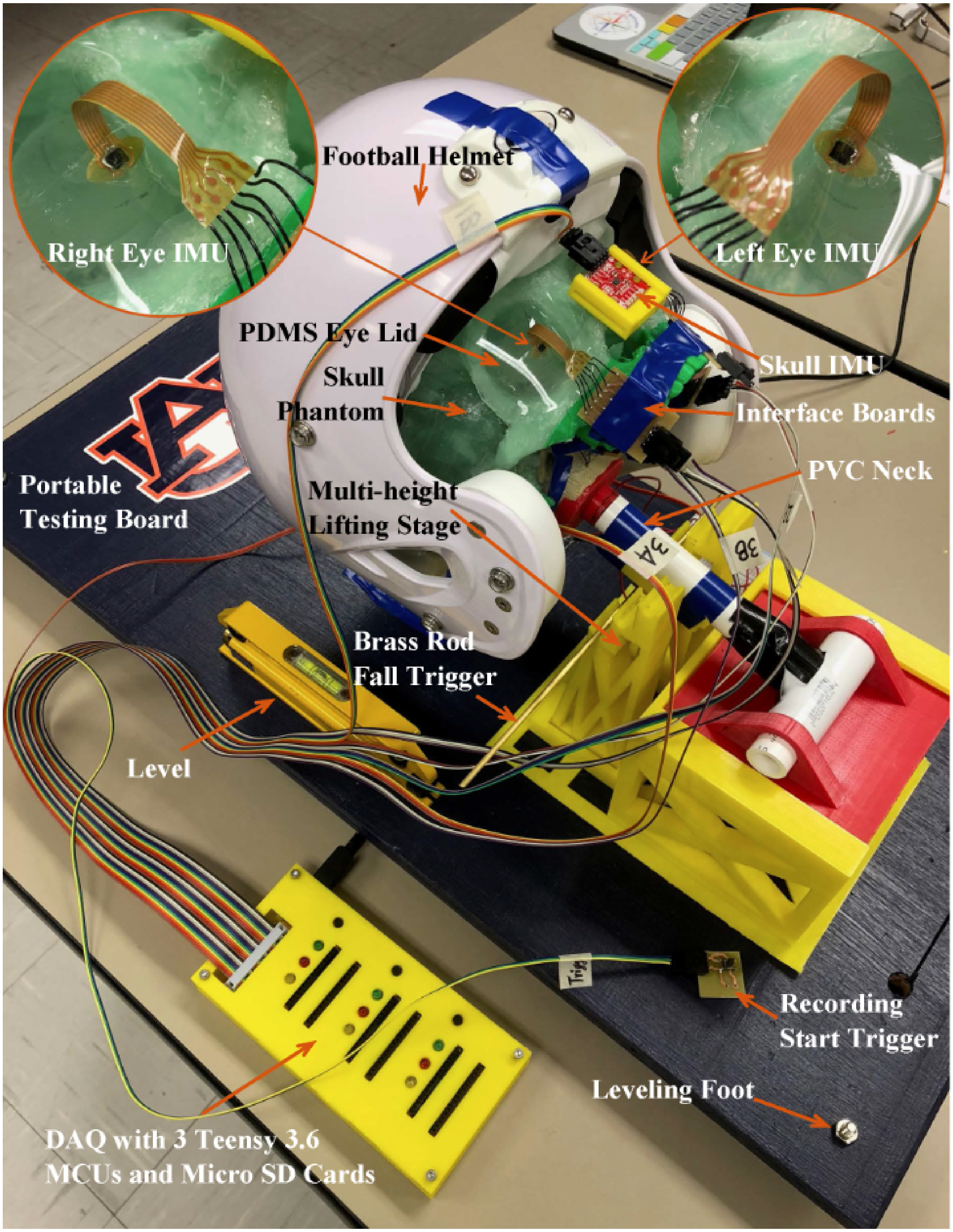


Figure 6.3: An Overview of the Test Apparatus.

by 1 mm thick PDMS eyelids. Gelatin of 4% and 2% mass ratios of gelatin to water were used for the emulation of the right and the left socket tissue respectively. A football helmet was put on to better emulate the real-world situation. Drops were initiated by removing the brass rod supporting the neck, allowing the skull and helmet to fall and collide with the platform surface. Sensing data were acquired by the Teensy™ DAQ introduced in Section 5.1 and logged in SD cards for further processing in MATLAB.

6.2 Data Processing and Analysis

A number of drop-and-impact experiments on this skull-brain-eye system were performed with the testing apparatus. Relative accelerations of the eye and brain IMU's to the skull along the impact direction were calculated applying the data fusion technique developed in Section 4.2.3. Instead of relative displacements, only relative accelerations were computed to further diminish errors introduced by integration. Figure 6.4 plots the relative accelerations from a typical result, where a vertical green solid line marks the impact time, a dashed blue line marks the 20 ms after the impact, and a red dashed line marks the impact end time. Since PER happens in approximately 20 ms after impact, the data within such periods were studied by polynomial regression analysis to find correlations (Appendix B.6). R^2 values for 5th order polynomial regressions of the data plotted in Figure 6.4 were calculated and recorded in Table 6.1. Since the R^2 values are coefficients of determination which "provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model" [137], the closer they are to one, the larger portion of data can be expressed by the regression model, and the better two studied accelerations are correlated. Notice that, even though the relative accelerations of the three brain IMU's appear to be quite different from those of the two eye IMU's over a longer time (Figure 6.4), they have a stronger correlation when zoomed in to the time window of 20 ms after impact which we consider as the time for PER, than the correlation of the skull and brain has (Table 6.1). In this case, the strongest correlation

IMU Positions	R^2 Values for 5th Order Polynomial Regression		
	Front Brain	Middle Brain	Posterior Brain
Right Eye	0.448	0.610	0.842
Left Eye	0.532	0.753	0.983
Skull	0.389	0.207	0.176

Table 6.1: Polynomial Regression Analysis of a Typical Test Data Set.

happened to be between the left eye and the posterior brain, with an R^2 of 0.983 in the linear regression analysis, while the weakest correlation was between the skull and the posterior brain with an R^2 of only 0.176.

The R^2 results from 18 repeatable tests including the one studied above are listed in Table 6.2 with their averages and standard deviations. The averages are visualized in Figure 6.5. Frequency spectra are plotted as histograms in Figure 6.6 with dashed blue lines marking the mean values and solid blue lines showing the beta distribution fit [138]. All the plots indicate that the movements of both eyes are better correlated to the brain than those of the skull, among which the left eye and the posterior brain have the best match, followed by the right eye and the posterior brain. Not only the LE-PB correlation is high in R^2 value, the concentrated distribution also reflects good repeatability of the experimental procedure.

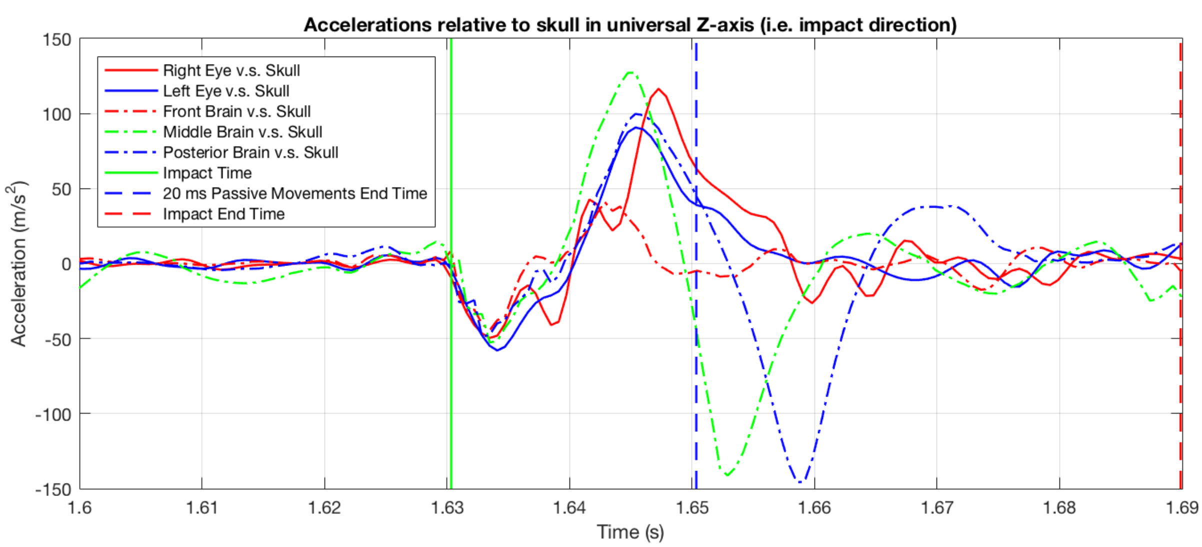


Figure 6.4: Relative Accelerations to the Skull in the Direction of Impact.

Data Set Number	R^2 Values for 5th Order Polynomial Regression								
	Right Eye			Left Eye			Skull		
	FB	MB	PB	FB	MB	PB	FB	MB	PB
1	0.196	0.798	0.955	0.095	0.973	0.968	0.836	0.365	0.402
2	0.461	0.861	0.938	0.585	0.942	0.970	0.704	0.310	0.381
3	0.458	0.457	0.965	0.480	0.423	0.963	0.479	0.500	0.299
4	0.975	0.980	0.991	0.946	0.987	0.991	0.540	0.540	0.527
5	0.859	0.358	0.873	0.842	0.604	0.993	0.685	0.206	0.268
6	0.404	0.693	0.900	0.647	0.920	0.989	0.793	0.092	0.077
7	0.654	0.399	0.779	0.807	0.597	0.961	0.081	0.082	0.245
8	0.448	0.610	0.842	0.532	0.753	0.983	0.389	0.207	0.176
9	0.482	0.560	0.510	0.549	0.898	0.800	0.774	0.431	0.459
10	0.394	0.754	0.875	0.573	0.930	0.990	0.837	0.113	0.149
11	0.697	0.819	0.959	0.805	0.915	0.991	0.893	0.281	0.390
12	0.463	0.525	0.885	0.638	0.711	0.976	0.321	0.095	0.083
13	0.598	0.774	0.973	0.552	0.878	0.993	0.608	0.193	0.216
14	0.779	0.868	0.973	0.795	0.845	0.991	0.551	0.594	0.392
15	0.386	0.935	0.975	0.552	0.944	0.972	0.921	0.160	0.224
16	0.900	0.908	0.980	0.895	0.901	0.995	0.887	0.363	0.388
17	0.683	0.796	0.925	0.624	0.935	0.991	0.948	0.249	0.098
18	0.483	0.828	0.960	0.529	0.904	0.991	0.797	0.060	0.231
Average	0.573	0.718	0.903	0.636	0.837	0.973	0.669	0.269	0.278
STD	0.208	0.189	0.113	0.197	0.156	0.045	0.237	0.166	0.134

Table 6.2: R^2 Values for 5th Order Polynomial Regression.

In contrary, the correlations between the brain parts and the skull are very week and rather random in distribution, which may explain the disability of helmet/mouthguard mounted sensors in on-field objective assessment of mTBI discussed in Section 2.3 and Section 3.1.

Notice the interesting fact that the mass ratios of gelatin used in LE, RE and PB were 2%, 4% and 4% respectively. With most of the other parameters identical, LE-PB had stronger correlation than the RE-PB did, even though RE and PB had the same concentration of gelatin. Since all the three IMU's were located closer to the inner wall of the skull/socket where the impact power was transmitted faster and earlier than to FB and MB. It is possible that this is a result of similar natural mechanical frequencies of LE and PB: one in an environment with less stiff gelatin (2%) but smaller space (eye socket), the other with stiffer gelatin (4%) yet larger space (skull cavity). Although the exact causation of this phenomenon is unknown, both eye accelerations were well correlated to the brain parts accelerations. The results thus suggest that sensing the passive eye accelerations (PER) using IMU's could provide better outcomes than sensing the skull (HACC) for real-time on-field objective concussion monitoring and diagnosis.

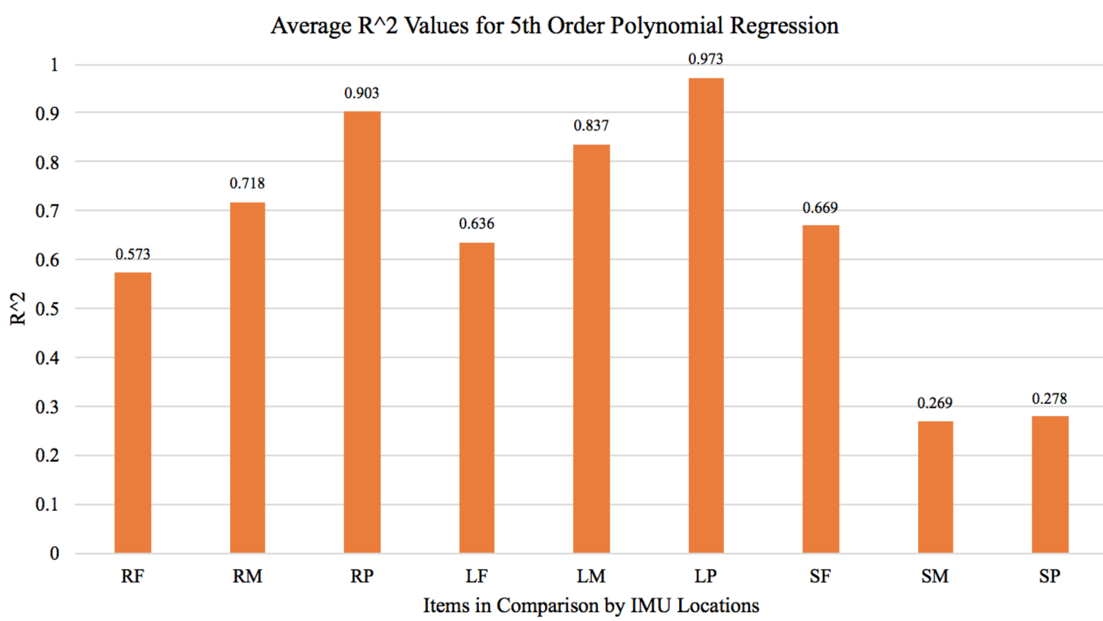


Figure 6.5: Average R^2 Values for 5th Order Polynomial Regression.

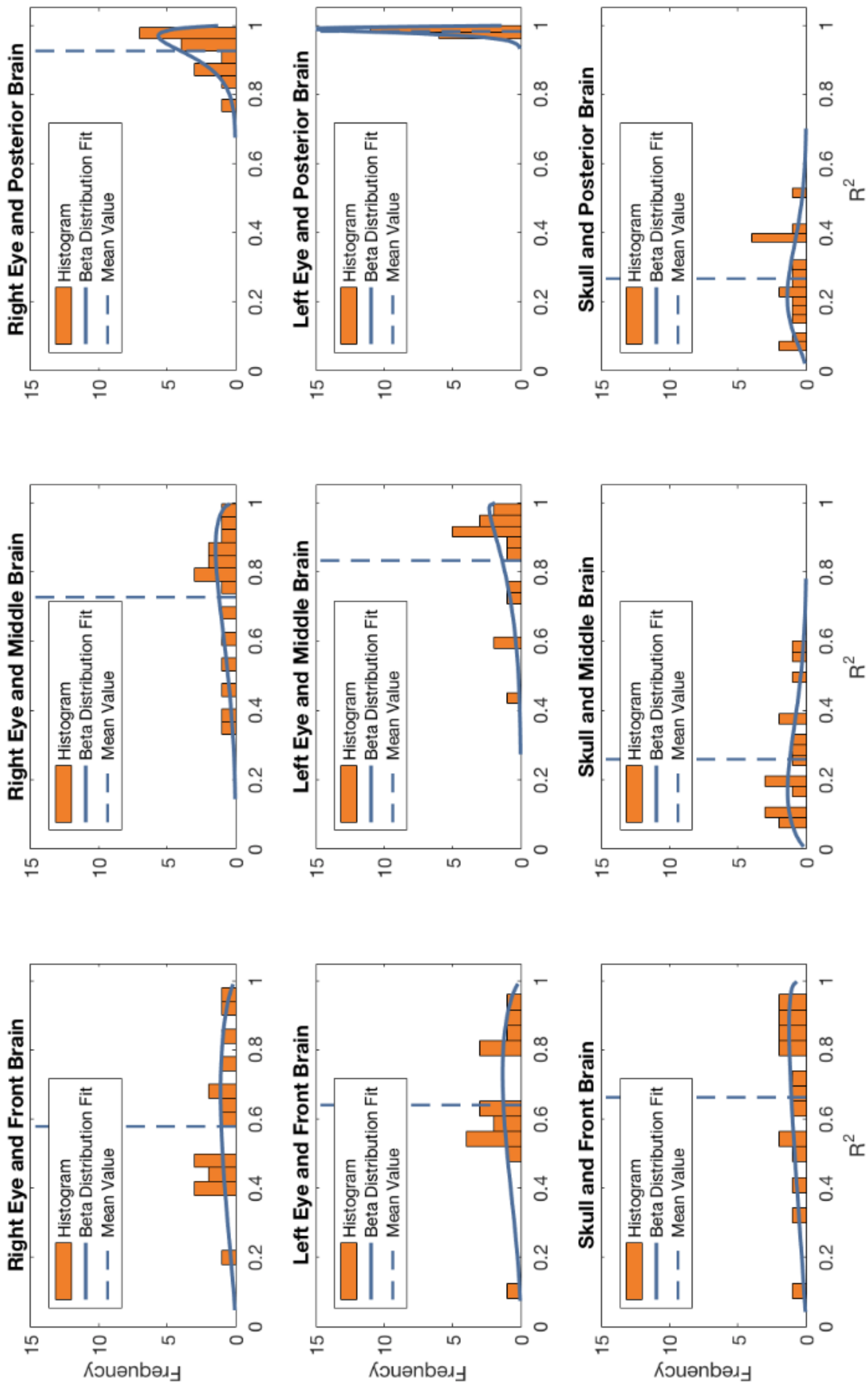


Figure 6.6: Histograms of R^2's and Beta Distribution Fits.

Chapter 7

Summary and Conclusion

mTBI/concussion, caused by the displacement and deformation of the brain in the skull during head/body impacts, is a high prevalence injury, especially among athletes in contact/fast-speed sports at all levels. The misdiagnosis of mTBI is often due to the subjective, inaccurate, post-event diagnosis techniques, and lack of self-reporting by the patients. It is thus vitally important to identify athletes who have experienced potentially concussive blows objectively real-time on the field in order to prevent further injury and long-term neurological damage. For this purpose, significant research has been conducted to on-field monitoring of HACC following the hypothesis that a high HACC is associated with a high risk of mTBI. However, this hypothesis has not been verified as it is not clear how HACC translates to brain displacement and deformation. Also, there is no method that can directly assess brain responses to head/body impacts in live humans who are at risk of mTBI at this moment.

According to previous research, it as been demonstrated that the eye and its surrounding tissue undergo passive mechanical linear and angular accelerations in the socket in a short time window immediately after HACC. Such passive accelerations are not neurally controlled, therefore it is hypothesized that a direct assessment of PER to HACC can be used to better infer brain response to impact, and consequently, can offer a better chance to predict mTBI than HACC itself.

Inspired by this hypothesis, in this research, a novel concept of using MEMS inertial sensors to record the acceleration of the eye as a surrogate to sense the acceleration of the brain during impacts was introduced and verified. Experimenting on 3D-printed socket models and on human volunteers, MEMS accelerometers was initially proved to be able to capture the relative displacement of the eye to the bony socket/skull in a similar fashion to the brain. Expansion to dynamic systems was achieved by applying 6-DoF IMU's, which were tested on a skull-brain-eye model and human volunteers in drop-and-impact experiments. An advanced sensor fusion technique was designed and applied in processing the IMU data. Similar angular accelerations of eye and brain relative to skull were observed in the IMU captured data in rotation tests on a VOR chair. Strong correlations of eye and brain accelerations were finally discovered in the drop-and-impact model tests thus suggest that sensing the PER using IMU's could provide better outcomes than sensing HACC for real-time on-field objective mTBI monitoring, assessment, and diagnosis.

Chapter 8

Future Work

The verification for applying MEMS inertial sensors in sensing PER as a surrogate for brain response to HACC for on-field objective assessment of concussion has been discussed in this dissertation as novel research. However, there is still a gap between the verified theory and a mature product. Challenges such as comparing the IMU data with tagged MRI data, and miniaturizing the sensors to a wireless contact lens need solutions. These tasks are beyond the scope of this dissertation and expected to be accomplished in future phases of relevant research.

Bibliography

[1] L. Bailey, "New concussion laws result in big jump in concussion treatment." [Online]. Available: http://ns.umich.edu/new/releases/22586-new-concussion-laws-result-in-big-jump-in-concussion-treatment

[2] S. L. Zuckerman, Z. Y. Kerr, A. Yengo-Kahn, E. Wasserman, T. Covassin, and G. S. Solomon, "Epidemiology of sports-related concussion in NCAA athletes from 2009-2010 to 2013-2014," *The American Journal of Sports Medicine*, vol. 43, no. 11, pp. 2654–2662, 2015.

[3] J. A. Langlois, W. Rutland-Brown, and M. M. Wald, "The epidemiology and impact of traumatic brain injury: a brief overview," *The Journal of head trauma rehabilitation*, vol. 21, no. 5, pp. 375–378, 2006.

[4] "Is a concussion the same as a mild traumatic brain injury?." School of Medicine, the University of Alabama at Birmingham, 2018. [Online]. Available: https://www.uab.edu/medicine/tbi/newly-injured/questions-about-traumatic-brain-injury-tbi/is-a-concussion-the-same-as-a-mild-tbi

[5] M. Faul, M. M. Wald, L. Xu, and V. G. Coronado, "Traumatic brain injury in the United States; emergency department visits, hospitalizations, and deaths, 2002-2006," Centers for Disease Control and Prevention, National Center for Injury Prevention and Control, Atlanta, GA, USA, 2010.

[6] E. A. Zerhouni, D. M. Parish, W. J. Rogers, A. Yang, and E. P. Shapiro, "Human heart: tagging with mr imaging–a method for noninvasive assessment of myocardial motion." *Radiology*, vol. 169, no. 1, pp. 59–63, 1988.

[7] L. Axel and L. Dougherty, "Mr imaging of motion with spatial modulation of magnetization." *Radiology*, vol. 171, no. 3, pp. 841–845, 1989.

[8] P. V. Bayly, E. H. Clayton, and G. M. Genin, "Quantitative imaging methods for the development and validation of brain biomechanics models," *Annual review of biomedical engineering*, vol. 14, pp. 369–396, 2012.

[9] J. Huch, "Concussion: A condition for audiology awareness." [Online]. Available: http://hearinghealthmatters.org/theaudiologycondition/2016/concussion-condition-audiology-awareness

[10] J. Gwin, J. Chu, T. McAllister, and R. Greenwald, “In situ measures of head impact acceleration in ncaa division i men’s ice hockey: implications for astm f1045 and other ice hockey helmet standards,” in *Fifth International Symposium on Safety in Ice Hockey*. ASTM International, 2009.

[11] J. G. Beckwith, R. M. Greenwald, and J. J. Chu, “Measuring head kinematics in football: correlation between the head impact telemetry system and hybrid iii headform,” *Annals of biomedical engineering*, vol. 40, no. 1, pp. 237–248, 2012.

[12] R. Jadischke, D. C. Viano, N. Dau, A. I. King, and J. McCarthy, “On the accuracy of the head impact telemetry (hit) system used in football helmets,” *Journal of biomechanics*, vol. 46, no. 13, pp. 2310–2315, 2013.

[13] M. A. Allison, Y. S. Kang, J. H. Bolte, M. R. Maltese, K. B. Arbogast *et al.*, “Validation of a helmet-based system to measure head impact biomechanics in ice hockey.” *Medicine and science in sports and exercise*, vol. 46, no. 1, pp. 115–123, 2014.

[14] K. M. Guskiewicz, J. P. Mihalik, V. Shankar, S. W. Marshall, D. H. Crowell, S. M. Oliaro, M. F. Ciocca, and D. N. Hooker, “Measurement of head impacts in collegiate football playersrelationship between head impact biomechanics and acute clinical outcome after concussion,” *Neurosurgery*, vol. 61, no. 6, pp. 1244–1253, 2007.

[15] M. A. McCaffrey, J. P. Mihalik, D. H. Crowell, E. W. Shields, and K. M. Guskiewicz, “Measurement of head impacts in collegiate football players: clinical measures of concussion after high-and low-magnitude impacts,” *Neurosurgery*, vol. 61, no. 6, pp. 1236–1243, 2007.

[16] S. P. Broglio, J. T. Eckner, T. Surma, and J. S. Kutcher, “Post-concussion cognitive declines and symptomatology are not related to concussion biomechanics in high school football players,” *Journal of neurotrauma*, vol. 28, no. 10, pp. 2061–2068, 2011.

[17] E. L. Breedlove, M. Robinson, T. M. Talavage, K. E. Morigaki, U. Yoruk, K. O’Keefe, J. King, L. J. Leverenz, J. W. Gilger, and E. A. Nauman, “Biomechanical correlates of symptomatic and asymptomatic neurophysiological impairment in high school football,” *Journal of biomechanics*, vol. 45, no. 7, pp. 1265–1272, 2012.

[18] J. R. Funk, S. Rowson, R. W. Daniel, and S. M. Duma, “Validation of concussion risk curves for collegiate football players derived from hits data,” *Annals of biomedical engineering*, vol. 40, no. 1, pp. 79–89, 2012.

[19] B. D. Jordan, “The clinical spectrum of sport-related traumatic brain injury,” *Nature Reviews Neurology*, vol. 9, no. 4, p. 222, 2013.

[20] T. M. Talavage, E. A. Nauman, E. L. Breedlove, U. Yoruk, A. E. Dye, K. E. Morigaki, H. Feuer, and L. J. Leverenz, “Functionally-detected cognitive impairment in high school football players without clinically-diagnosed concussion,” *Journal of neurotrauma*, vol. 31, no. 4, pp. 327–338, 2014.

[21] H. Collewijn and J. B. Smeets, "Early components of the human vestibulo-ocular response to head rotation: latency and gain," *Journal of Neurophysiology*, vol. 84, no. 1, pp. 376–389, 2000.

[22] V. G. Coronado, L. C. McGuire, M. Faul, D. E. Sugerman, and W. S. Pearson, "Traumatic brain injury epidemiology and public health issues," *Brain injury medicine: Principles and practice*, vol. 84, 2012.

[23] N. D. Zasler, D. I. Katz, and R. D. Zafonte, *Brain injury medicine: principles and practice*. Demos Medical Publishing, 2013.

[24] S. Physique, "Traumatic brain injury." [Online]. Available: http://www.santephysique.com/blog/traumatic-brain-injury

[25] M. E. Halstead, K. D. Walter *et al.*, "Sport-related concussion in children and adolescents," *Pediatrics*, vol. 126, no. 3, pp. 597–615, 2010.

[26] C. for Disease Control, Prevention *et al.*, "Nonfatal traumatic brain injuries from sports and recreation activities–united states, 2001-2005," *MMWR: Morbidity and mortality weekly report*, vol. 56, no. 29, pp. 733–737, 2007.

[27] W. P. Meehan III, R. C. Mannix, M. J. O'brien, and M. W. Collins, "The prevalence of undiagnosed concussions in athletes," *Clinical journal of sport medicine: official journal of the Canadian Academy of Sport Medicine*, vol. 23, no. 5, p. 339, 2013.

[28] K. B. Arbogast, A. D. McGinley, C. L. Master, M. F. Grady, R. L. Robinson, and M. R. Zonfrillo, "Cognitive rest and school-based recommendations following pediatric concussion: the need for primary care support tools," *Clinical pediatrics*, vol. 52, no. 5, pp. 397–402, 2013.

[29] R. C. Cantu, "Second-impact syndrome," *Clinics in sports medicine*, vol. 17, no. 1, pp. 37–44, 1998.

[30] K. M. Guskiewicz, S. W. Marshall, J. Bailes, M. McCrea, R. C. Cantu, C. Randolph, and B. D. Jordan, "Association between recurrent concussion and late-life cognitive impairment in retired professional football players," *Neurosurgery*, vol. 57, no. 4, pp. 719–726, 2005.

[31] K. M. Guskiewicz, S. W. Marshall, J. Bailes, M. McCrea, H. P. Harding, A. Matthews, J. R. Mihalik, and R. C. Cantu, "Recurrent concussion and risk of depression in retired professional football players," *Medicine and science in sports and exercise*, vol. 39, no. 6, p. 903, 2007.

[32] B. I. Omalu, R. P. Fitzsimmons, J. Hammers, and J. Bailes, "Chronic traumatic encephalopathy in a professional american wrestler," *Journal of forensic nursing*, vol. 6, no. 3, pp. 130–136, 2010.

[33] B. D. Jordan, "Chronic traumatic brain injury associated with boxing," in *Seminars in neurology*, vol. 20, no. 02. Copyright© 2000 by Thieme Medical Publishers, Inc., 333 Seventh Avenue, New York, NY 10001, USA. Tel.:+ 1 (212) 584-4662, 2000, pp. 179–186.

[34] D. H. Daneshvar, D. O. Riley, C. J. Nowinski, A. C. McKee, R. A. Stern, and R. C. Cantu, "Long-term consequences: effects on normal development profile after concussion," *Physical Medicine and Rehabilitation Clinics*, vol. 22, no. 4, pp. 683–700, 2011.

[35] B. L. Plassman, R. Havlik, D. Steffens, M. Helms, T. Newman, D. Drosdick, C. Phillips, B. Gau, K. Welsh-Bohmer, J. Burke *et al.*, "Documented head injury in early adulthood and risk of alzheimers disease and other dementias," *Neurology*, vol. 55, no. 8, pp. 1158–1166, 2000.

[36] A. C. McKee, R. C. Cantu, C. J. Nowinski, E. T. Hedley-Whyte, B. E. Gavett, A. E. Budson, V. E. Santini, H.-S. Lee, C. A. Kubilus, and R. A. Stern, "Chronic traumatic encephalopathy in athletes: progressive tauopathy after repetitive head injury," *Journal of Neuropathology & Experimental Neurology*, vol. 68, no. 7, pp. 709–735, 2009.

[37] H. Chen, M. Richard, D. P. Sandler, D. M. Umbach, and F. Kamel, "Head injury and amyotrophic lateral sclerosis," *American journal of epidemiology*, vol. 166, no. 7, pp. 810–816, 2007.

[38] P. McCrory, K. Johnston, W. Meeuwisse, M. Aubry, R. Cantu, J. Dvorak, T. Graf-Baumann, J. Kelly, M. Lovell, and P. Schamasch, "Summary and agreement statement of the 2nd international conference on concussion in sport, prague 2004," *British journal of sports medicine*, vol. 39, no. suppl 1, pp. i78–i86, 2005.

[39] A. Ommaya, W. Goldsmith, and L. Thibault, "Biomechanics and neuropathology of adult and paediatric head injury," *British journal of neurosurgery*, vol. 16, no. 3, pp. 220–242, 2002.

[40] A. K. Ommaya, R. L. Grubb Jr, and R. A. Naumann, "Coup and contre-coup injury: observations on the mechanics of visible brain injuries in the rhesus monkey," *Journal of neurosurgery*, vol. 35, no. 5, pp. 503–516, 1971.

[41] A. Holbourn, "Mechanics of head injuries," *The Lancet*, vol. 242, no. 6267, pp. 438–441, 1943.

[42] D. F. Meaney, D. H. Smith, D. I. Shreiber, A. C. Bain, R. T. Miller, D. T. Ross, and T. A. Gennarelli, "Biomechanical analysis of experimental diffuse axonal injury," *Journal of neurotrauma*, vol. 12, no. 4, pp. 689–694, 1995.

[43] S. S. Margulies, L. E. Thibault, and T. A. Gennarelli, "Physical model simulations of brain injury in the primate," *Journal of biomechanics*, vol. 23, no. 8, pp. 823–836, 1990.

[44] W. Hardy, C. Foster, A. King, and S. Tashman, "Investigation of brain injury kinematics: introduction of a new technique," *ASME Applied Mechanics Division-Publications-AMD*, vol. 225, pp. 241–254, 1997.

[45] W. N. Hardy, C. D. Foster, M. J. Mason, K. H. Yang, A. I. King, and S. Tashman, "Investigation of head injury mechanisms using neutral density technology and high-speed biplanar x-ray." *Stapp car crash journal*, vol. 45, pp. 337–368, 2001.

[46] W. N. Hardy, M. J. Mason, C. D. Foster, C. S. Shah, J. M. Kopacz, K. H. Yang, A. I. King, J. Bishop, M. Bey, W. Anderst *et al.*, "A study of the response of the human cadaver head to impact," *Stapp car crash journal*, vol. 51, p. 17, 2007.

[47] V. Hodgson, E. Gurdjian, and L. Thomas, "Experimental skull deformation and brain displacement demonstrated by flash x-ray technique," *Journal of neurosurgery*, vol. 25, no. 5, pp. 549–552, 1966.

[48] H. Gosch, E. Gooding, and R. Schneider, "Distortion and displacement of the brain in experimental head injuries." in *Surgical forum*, vol. 20, 1969, p. 425.

[49] G. S. Nusholtz, P. Lux, P. Kaiker, and M. A. Janicki, "Head impact responseskull deformation and angular accelerations," SAE Technical Paper, Tech. Rep., 1984.

[50] R. H. Pudenz and C. H. Shelden, "The lucite calvariuma method for direct observation of the brain: Ii. cranial trauma and brain movement," *Journal of neurosurgery*, vol. 3, no. 6, pp. 487–505, 1946.

[51] N. G. Ibrahim, R. Natesh, S. E. Szczesny, K. Ryall, S. A. Eucker, B. Coats, and S. S. Margulies, "In situ deformations in the immature brain during rapid rotations," *Journal of biomechanical engineering*, vol. 132, no. 4, p. 044501, 2010.

[52] Y. Feng, T. M. Abney, R. J. Okamoto, R. B. Pless, G. M. Genin, and P. V. Bayly, "Relative brain displacement and deformation during constrained mild frontal head impact," *Journal of the Royal Society Interface*, vol. 7, no. 53, pp. 1677–1688, 2010.

[53] S. Ji, Q. Zhu, L. Dougherty, and S. S. Margulies, "In vivo measurements of human brain displacement," *Stapp car crash journal*, vol. 48, p. 227, 2004.

[54] S. Ji and S. S. Margulies, "In vivo pons motion within the skull," *Journal of biomechanics*, vol. 40, no. 1, pp. 92–99, 2007.

[55] A. A. Sabet, E. Christoforou, B. Zatlin, G. M. Genin, and P. V. Bayly, "Deformation of the human brain induced by mild angular head acceleration," *Journal of biomechanics*, vol. 41, no. 2, pp. 307–315, 2008.

[56] J. Ho and S. Kleiven, "Dynamic response of the brain with vasculature: a three-dimensional computational study," *Journal of biomechanics*, vol. 40, no. 13, pp. 3006–3012, 2007.

[57] D. C. Viano, I. R. Casson, E. J. Pellman, L. Zhang, A. I. King, and K. H. Yang, "Concussion in professional football: brain responses by finite element analysis: part 9," *Neurosurgery*, vol. 57, no. 5, pp. 891–916, 2005.

[58] L. Zhang, K. H. Yang, R. Dwarampudi, K. Omori, T. Li, K. Chang, W. N. Hardy, T. B. Khalil, and A. I. King, "Recent advances in brain injury research: a new human head model development and validation," *Stapp Car Crash J*, vol. 45, no. 11, pp. 369–394, 2001.

[59] L. Zhang, K. H. Yang, and A. I. King, "Comparison of brain responses between frontal and lateral impacts by finite element modeling," *Journal of neurotrauma*, vol. 18, no. 1, pp. 21–30, 2001.

[60] ——, "A proposed injury threshold for mild traumatic brain injury," *Journal of biomechanical engineering*, vol. 126, no. 2, pp. 226–236, 2004.

[61] H. Mao, L. Zhang, B. Jiang, V. V. Genthikatti, X. Jin, F. Zhu, R. Makwana, A. Gill, G. Jandir, A. Singh *et al.*, "Development of a finite element human head model partially validated with thirty five experimental cases," *Journal of biomechanical engineering*, vol. 135, no. 11, p. 111002, 2013.

[62] M. McCrea, T. Hammeke, G. Olsen, P. Leo, and K. Guskiewicz, "Unreported concussion in high school football players: implications for prevention," *Clinical journal of sport medicine*, vol. 14, no. 1, pp. 13–17, 2004.

[63] K. M. Guskiewicz and J. P. Mihalik, "Biomechanics of sport concussion: quest for the elusive injury threshold," *Exercise and sport sciences reviews*, vol. 39, no. 1, pp. 4–11, 2011.

[64] C. L. Ewing, D. J. Thomas, G. W. Beeler Jr, L. M. Patrick, and D. B. Gillis, "Dynamic response of the head and neck of the living human to-gx impact acceleration. 1. experimental design and preliminary experimental data," NAVAL AEROSPACE MEDICAL INST PENSACOLA FL, Tech. Rep., 1969.

[65] C. Withnall, N. Shewchenko, M. Wonnacott, and J. Dvorak, "Effectiveness of headgear in football," *British journal of sports medicine*, vol. 39, no. suppl 1, pp. i40–i48, 2005.

[66] T. P. Ng, W. R. Bussone, and S. M. Duma, "The effect of gender and body size on linear accelerations of the head observed during daily activities." *Biomedical sciences instrumentation*, vol. 42, pp. 25–30, 2006.

[67] W. Bussone, "Linear and angular head accelerations in daily life," Ph.D. dissertation, Virginia Tech, 2005.

[68] M. E. Allen, I. Weir-Jones, D. R. Motiuk, K. R. Flewin, R. D. Goring, R. Kobetitch, and A. Broadhurst, "Acceleration perturbations of daily living. a comparison to'whiplash'." *Spine*, vol. 19, no. 11, pp. 1285–1290, 1994.

[69] R. S. Naunheim, J. Standeven, C. Richter, and L. M. Lewis, "Comparison of impact data in hockey, football, and soccer," *Journal of Trauma and Acute Care Surgery*, vol. 48, no. 5, pp. 938–941, 2000.

[70] J. R. Funk, J. M. Cormier, C. E. Bain, H. Guzman, E. Bonugli, and S. J. Manoogian, "Head and neck loading in everyday and vigorous activities," *Annals of biomedical engineering*, vol. 39, no. 2, pp. 766–776, 2011.

[71] Simbex, "Hit system." [Online]. Available: https://simbex.com/work/riddell

[72] Riddell, "Riddel iq made up of riddell srs and riddell insite." [Online]. Available: http://www.riddell.com/riddell-iq

[73] S. P. Broglio, J. J. Sosnoff, S. Shin, X. He, C. Alcaraz, and J. Zimmerman, "Head impacts during high school football: a biomechanical assessment," *Journal of athletic training*, vol. 44, no. 4, pp. 342–349, 2009.

[74] S. P. Broglio, J. T. Eckner, D. Martini, J. J. Sosnoff, J. S. Kutcher, and C. Randolph, "Cumulative head impact burden in high school football," *Journal of neurotrauma*, vol. 28, no. 10, pp. 2069–2078, 2011.

[75] P. G. Brolinson, S. Manoogian, D. McNeely, M. Goforth, R. Greenwald, and S. Duma, "Analysis of linear head accelerations from collegiate football impacts," *Current sports medicine reports*, vol. 5, no. 1, pp. 23–28, 2006.

[76] J. F. Wisniewski, K. Guskiewicz, M. Trope, and A. Sigurdsson, "Incidence of cerebral concussions associated with type of mouthguard used in college football," *Dental traumatology*, vol. 20, no. 3, pp. 143–149, 2004.

[77] A. Bartsch, S. Samorezov, E. Benzel, V. Miele, and D. Brett, "Validation of an intelligent mouthguard single event head impact dosimeter," SAE Technical Paper, Tech. Rep., 2014.

[78] D. King, P. A. Hume, M. Brughelli, and C. Gissane, "Instrumented mouthguard acceleration analyses for head impacts in amateur rugby union players over a season of matches," *The American journal of sports medicine*, vol. 43, no. 3, pp. 614–624, 2015.

[79] PreventBiometrics. [Online]. Available: http://preventbiometrics.com

[80] M. H. Heitger, R. D. Jones, A. Macleod, D. L. Snell, C. M. Frampton, and T. J. Anderson, "Impaired eye movements in post-concussion syndrome indicate suboptimal brain function beyond the influence of depression, malingering or intellectual ability," *Brain*, vol. 132, no. 10, pp. 2850–2870, 2009.

[81] K. J. Ciuffreda, D. Ludlam, and P. Thiagarajan, "Oculomotor diagnostic protocol for the mtbi population," 2011.

[82] J. E. Capó-Aponte, A. K. Tarbett, T. G. Urosevich, L. A. Temme, N. K. Sanghera, and M. E. Kalich, "Effectiveness of computerized oculomotor vision screening in a military population: Pilot study," ARMY AEROMEDICAL RESEARCH LAB FORT RUCKER AL, Tech. Rep., 2012.

[83] R. E. Ventura, L. J. Balcer, and S. L. Galetta, "The neuro-ophthalmology of head trauma," *The Lancet Neurology*, vol. 13, no. 10, pp. 1006–1016, 2014.

[84] U. Samadani, R. Ritlop, M. Reyes, E. Nehrbass, M. Li, E. Lamm, J. Schneider, D. Shimunov, M. Sava, R. Kolecki *et al.*, "Eye tracking detects disconjugate eye movements associated with structural traumatic brain injury and concussion," *Journal of neurotrauma*, vol. 32, no. 8, pp. 548–556, 2015.

[85] EyeLink. [Online]. Available: https://www.sr-research.com/products/eyelink-1000-plus

[86] Z. Sharafi, Z. Soh, and Y.-G. Guéhéneuc, "A systematic literature review on the usage of eye-tracking in software engineering," *Information and Software Technology*, vol. 67, pp. 79–107, 2015.

[87] P. H. Donaldson, C. Gurvich, J. Fielding, and P. G. Enticott, "Exploring associations between gaze patterns and putative human mirror neuron system activity," *Frontiers in human neuroscience*, vol. 9, p. 396, 2015.

[88] S. Rowson and S. M. Duma, "The virginia tech response," *Annals of biomedical engineering*, vol. 40, no. 12, pp. 2512–2518, 2012.

[89] ImPACT Applications Inc. [Online]. Available: https://impacttest.com

[90] S. Rowson, S. M. Duma, J. G. Beckwith, J. J. Chu, R. M. Greenwald, J. J. Crisco, P. G. Brolinson, A.-C. Duhaime, T. W. McAllister, and A. C. Maerlender, "Rotational head kinematics in football impacts: an injury risk function for concussion," *Annals of biomedical engineering*, vol. 40, no. 1, pp. 1–13, 2012.

[91] S. Rowson, J. G. Beckwith, J. J. Chu, D. S. Leonard, R. M. Greenwald, and S. M. Duma, "A six degree of freedom head acceleration measurement device for use in football," *Journal of applied biomechanics*, vol. 27, no. 1, pp. 8–14, 2011.

[92] D. Graham, J. H. Adams, J. Nicoll, W. Maxwell, and T. Gennarelli, "The nature, distribution and causes of traumatic brain injury," *Brain Pathology*, vol. 5, no. 4, pp. 397–406, 1995.

[93] "Anatomy of eye socket orbital tumor cancer." [Online]. Available: http://iarekylew00t.me/anatomy-of-eye-socket/anatomy-of-eye-socket-orbital-tumor-cancer

[94] G. Bush and F. Miles, "Short-latency compensatory eye movements associated with a brief period of free fall," *Experimental brain research*, vol. 108, no. 2, pp. 337–340, 1996.

[95] V. reflex. [Online]. Available: https://en.wikipedia.org/wiki/Vestibuloocular_reflex

[96] J.-L. Vercher, G. Gauthier, E. Marchetti, P. Mandelbrojt, and Y. Ebihara, "Origin of eye movements induced by high frequency rotation of the head." *Aviation, space, and environmental medicine*, vol. 55, no. 11, pp. 1046–1050, 1984.

[97] U. Schwarz and F. Miles, "Ocular responses to translation and their dependence on viewing distance. i. motion of the observer," *Journal of neurophysiology*, vol. 66, no. 3, pp. 851–864, 1991.

[98] S. Tabak and H. Collewijn, "Human vestibulo-ocular responses to rapid, helmet-driven head movements," *Experimental brain research*, vol. 102, no. 2, pp. 367–378, 1994.

[99] S. Tabak, H. Collewijn, L. Boumans, and J. Van der Steen, "Gain and delay of human vestibulo-ocular reflexes to oscillation and steps of the head by a reactive torque helmet: I. normal subjects," *Acta oto-laryngologica*, vol. 117, no. 6, pp. 785–795, 1997.

[100] P. Jombik and V. Bahỳl, "Short latency responses in the averaged electro-oculogram elicited by vibrational impulse stimuli applied to the skull: could they reflect vestibulo-ocular reflex function?" *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 76, no. 2, pp. 222–228, 2005.

[101] L. B. Minor, D. M. Lasker, D. D. Backous, and T. E. Hullar, "Horizontal vestibuloocular reflex evoked by high-acceleration rotations in the squirrel monkey. i. normal responses," *Journal of Neurophysiology*, vol. 82, no. 3, pp. 1254–1270, 1999.

[102] H. N. Abramson, "The dynamic behavior of liquids in moving containers, with applications to space vehicle technology," 1966.

[103] H. C. Mayer and R. Krechetnikov, "Walking with coffee: Why does it spill?" *Physical Review E*, vol. 85, no. 4, p. 046117, 2012.

[104] S. Cirovic, R. Bhola, D. Hose, I. Howard, P. Lawford, and M. Parsons, "A computational study of the passive mechanisms of eye restraint during head impact trauma," *Computer methods in biomechanics and biomedical engineering*, vol. 8, no. 1, pp. 1–6, 2005.

[105] J. W. Miles, "On the sloshing of liquid in a cylindrical tank," Thompson Ramo Wooldridge Inc Los Angeles CA, Tech. Rep., 1956.

[106] A. Holbourn, "The mechanics of brain injuries," *British medical bulletin*, vol. 3, no. 6, pp. 147–149, 1945.

[107] D. W. Smith, J. E. Bailes, J. A. Fisher, J. Robles, R. C. Turner, and J. D. Mills, "Internal jugular vein compression mitigates traumatic axonal injury in a rat model by reducing the intracranial slosh effect," *Neurosurgery*, vol. 70, no. 3, pp. 740–746, 2011.

[108] R. C. Turner, Z. J. Naser, J. E. Bailes, D. W. Smith, J. A. Fisher, and C. L. Rosen, "Effect of slosh mitigation on histologic markers of traumatic brain injury," *Journal of neurosurgery*, vol. 117, no. 6, pp. 1110–1118, 2012.

[109] G. D. Myer, D. Smith, K. D. Barber Foss, C. A. Dicesare, A. W. Kiefer, A. M. Kushner, S. M. Thomas, H. Sucharew, and J. C. Khoury, "Rates of concussion are lower in national football league games played at higher altitudes," *journal of orthopaedic & sports physical therapy*, vol. 44, no. 3, pp. 164–172, 2014.

[110] S. Maguire, P. Watts, A. Shaw, S. Holden, R. Taylor, W. Watkins, M. Mann, V. Tempest, and A. Kemp, "Retinal haemorrhages and related findings in abusive and non-abusive head trauma: a systematic review," *Eye*, vol. 27, no. 1, p. 28, 2013.

[111] E. D. Weichel, M. H. Colyer, C. Bautista, K. S. Bower, and L. M. French, "Traumatic brain injury associated with combat ocular trauma," *The Journal of head trauma rehabilitation*, vol. 24, no. 1, pp. 41–50, 2009.

[112] M. O. Hughes, "A pictorial anatomy of the human eye/anophthalmic socket: a review for ocularists," *eye*, vol. 4, no. 5, p. 6, 2007.

[113] W. F. Cygan and W. J. Benjamin, "Features of the partially expanded human inferior conjunctival sac," *Acta Ophthalmologica*, vol. 73, no. 6, pp. 555–559, 1995.

[114] "ADXL325 datasheet." Analog Devices, 2009. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL325.pdf

[115] LPKF Laser and Electronics. [Online]. Available: https://www.lpkf.com/products/rapid-pcb-prototyping/circuit-board-plotter/protomat-s103.htm

[116] R. L. Burden and J. D. Faires, *Numerical analysis (7th)*. Prindle Weber and Schmidt, Boston, 2001.

[117] Y. Meng, M. L. Adams, L. Liu, and M. Bolding, "Application of MEMS accelerometers in sensing passive eye response as a surrogate for brain response to head acceleration," in *2016 IEEE SENSORS*, Oct 2016, pp. 1–3.

[118] MacGyver, "Sliced human skull with mandible and teeth." [Online]. Available: http://www.thingiverse.com/thing:43591

[119] I. Bekerman, P. Gottlieb, and M. Vaiman, "Variations in eyeball diameters of the healthy adults," *Journal of Ophthalmology*, vol. 2014, 2014.

[120] H. Davson, "Human eye." 2016. [Online]. Available: https://www.britannica.com/science/human-eye

[121] D. X. Cifu, J. R. Wares, K. W. Hoke, P. A. Wetzel, G. Gitchel, and W. Carne, "Differential eye movements in mild traumatic brain injury versus normal controls," *The Journal of head trauma rehabilitation*, vol. 30, no. 1, pp. 21–28, 2015.

[122] "LSM6DS3 datasheet." STMicroelectronics, 2016. [Online]. Available: www.st.com/resource/en/datasheet/lsm6ds3.pdf

[123] "DuPont Pyralux AC flexible circuit materials." DuPont, 2009. [Online]. Available: http://www.dupont.com/content/dam/dupont/products-and-services/electronic-and-electrical-materials/flexible-rigid-flex-circuit-materials/documents/PyraluxACclad-DataSheet.pdf

[124] R. Richardson, J. Miller, and W. Reichert, "Polyimides as biomaterials: preliminary biocompatibility testing," *Biomaterials*, vol. 14, no. 8, pp. 627–635, 1993.

[125] L. Brancato, G. Keulemans, T. Verbelen, B. Meyns, and R. Puers, "An implantable intravascular pressure sensor for a ventricular assist device," *Micromachines*, vol. 7, no. 8, p. 135, 2016.

[126] T. Belytschko, T. Andriacchi, A. Schultz, and J. Galante, "Analog studies of forces in the human spine: computational techniques," *Journal of Biomechanics*, vol. 6, no. 4, pp. 361–371, 1973.

[127] "SparkFun 6 degrees of freedom breakout - LSM6DS3." SparkFun, 2015. [Online]. Available: https://www.sparkfun.com/products/13339

[128] "Teensy USB board, version 3.6." PJRC.com, LLC. [Online]. Available: https://www.pjrc.com/store/teensy36.html

[129] V. Kempe, *Inertial MEMS: principles and practice*. Cambridge University Press, 2011.

[130] T. K. Nguyen, M. Ranieri, J. DiGiovanna, O. Peter, V. Genovese, A. P. Fornos, and S. Micera, "A real-time research platform to study vestibular implants with gyroscopic inputs in vestibular deficient subjects," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 474–484, 2014.

[131] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing uav," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 340–345.

[132] S. P. Tseng, W.-L. Li, C.-Y. Sheng, J.-W. Hsu, and C.-S. Chen, "Motion and attitude estimation using inertial measurements with complementary filter," in *Control Conference (ASCC), 2011 8th Asian*. IEEE, 2011, pp. 863–868.

[133] J. R. Taylor, *An introduction to error analysis: the study of uncertainties in physical measurements*. California, University Science Books, 1982.

[134] Y. Meng, B. Bottenfield, M. Bolding, L. Liu, and M. L. Adams, "Sensing passive eye response to impact induced head acceleration using mems imus," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 182–191, Feb 2018.

[135] "CC430F5137 datasheet." Texas Instruments, 2013. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc430f5135.pdf

[136] A. Deutsch, P. W. Coteus, G. V. Kopcsay, H. H. Smith, C. W. Surovic, B. L. Krauter, D. C. Edelstein, and P. L. Restle, "On-chip wiring design challenges for gigahertz operation," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 529–555, Apr 2001.

[137] "Coefficient of Determination." Wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Coefficient_of_determination

[138] "Beta Distribution." Wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Beta_distribution

Appendix A

PCB Layouts Designed in EAGLE CAD

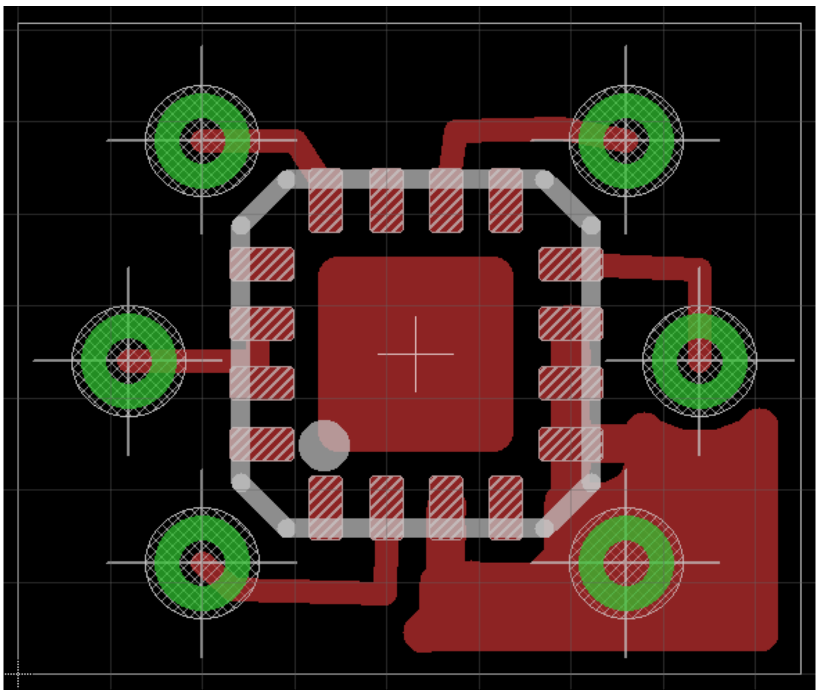


Figure A.1: The Layout for ADXL325 Accelerometer Rigid PCB (8.5 mm × 7 mm with 1-mm grid on).

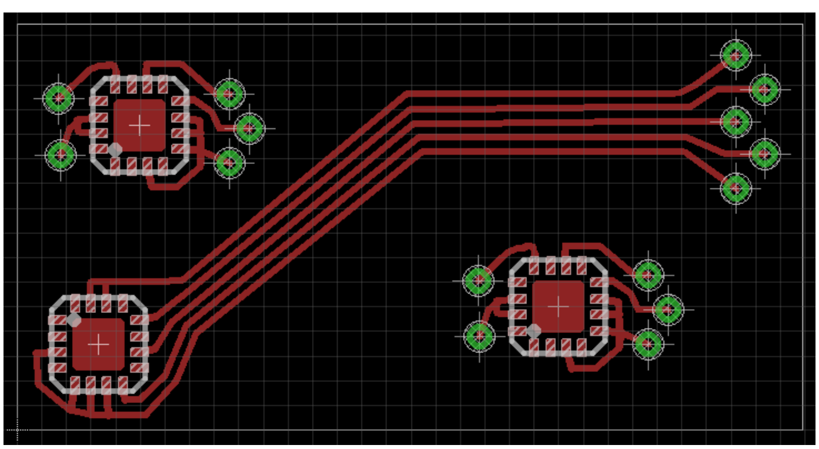


Figure A.2: The Layout for ADXL325 Accelerometer Flexible PCBs (with 1-mm grid on).

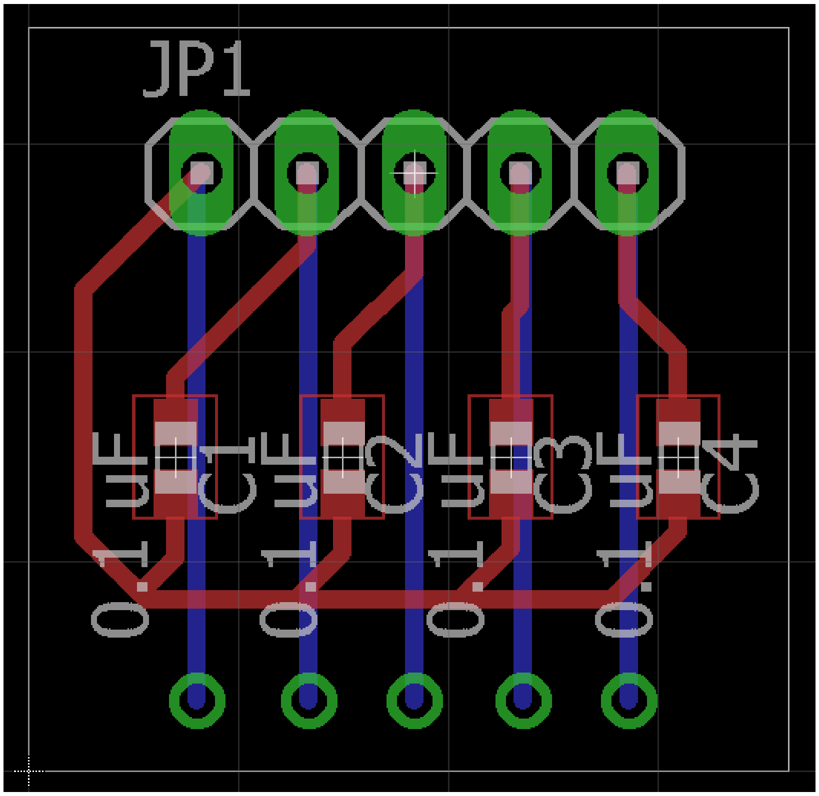


Figure A.3: The Layout for ADXL325 Accelerometer Interface PCB (1.8 cm × 1.7 cm with 5-mm grid on).

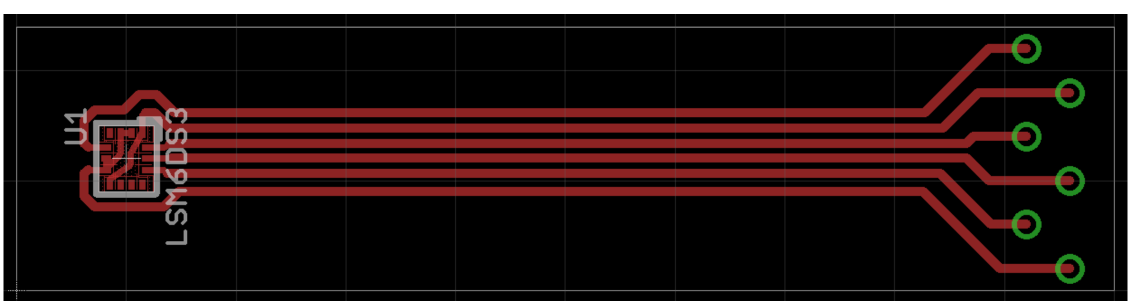


Figure A.4: The Layout for LSM6DS3 IMU Flexible Ribbon-style PCB (with 5-mm grid on).

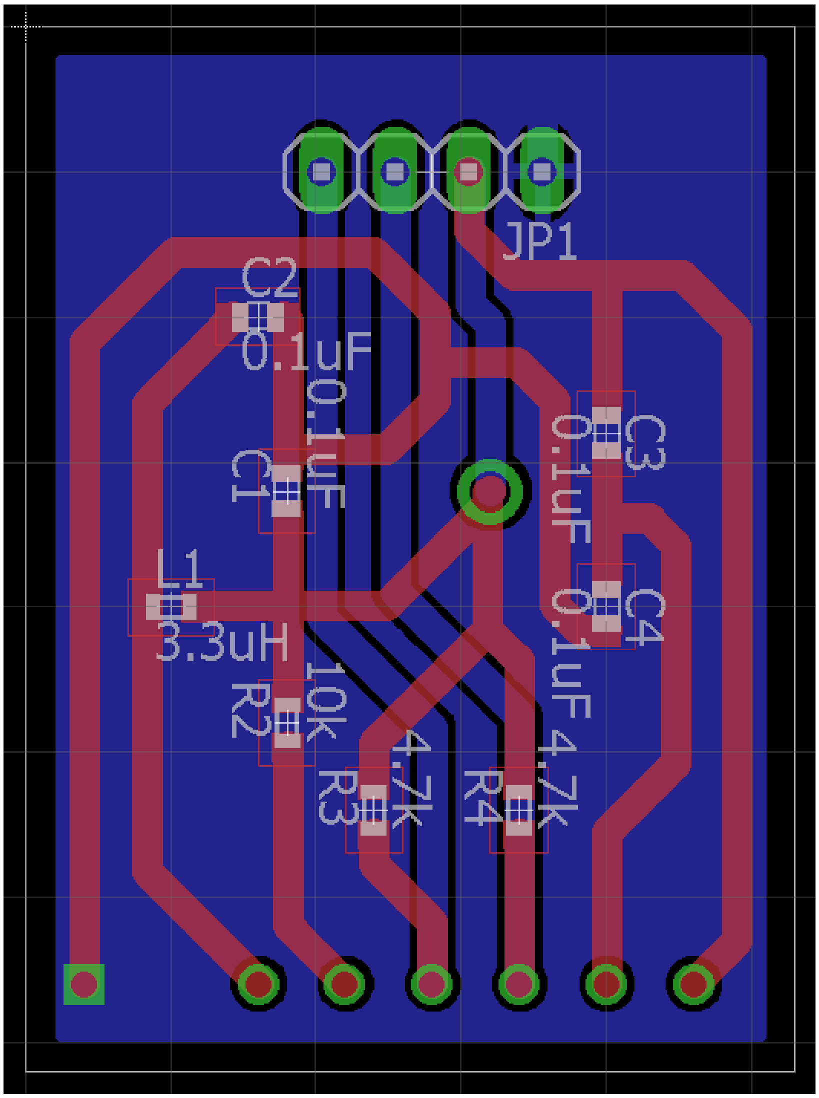


Figure A.5: The Layout for LSM6DS3 IMU Interface PCB (3.6 cm × 2.65 cm with 5-mm grid on).

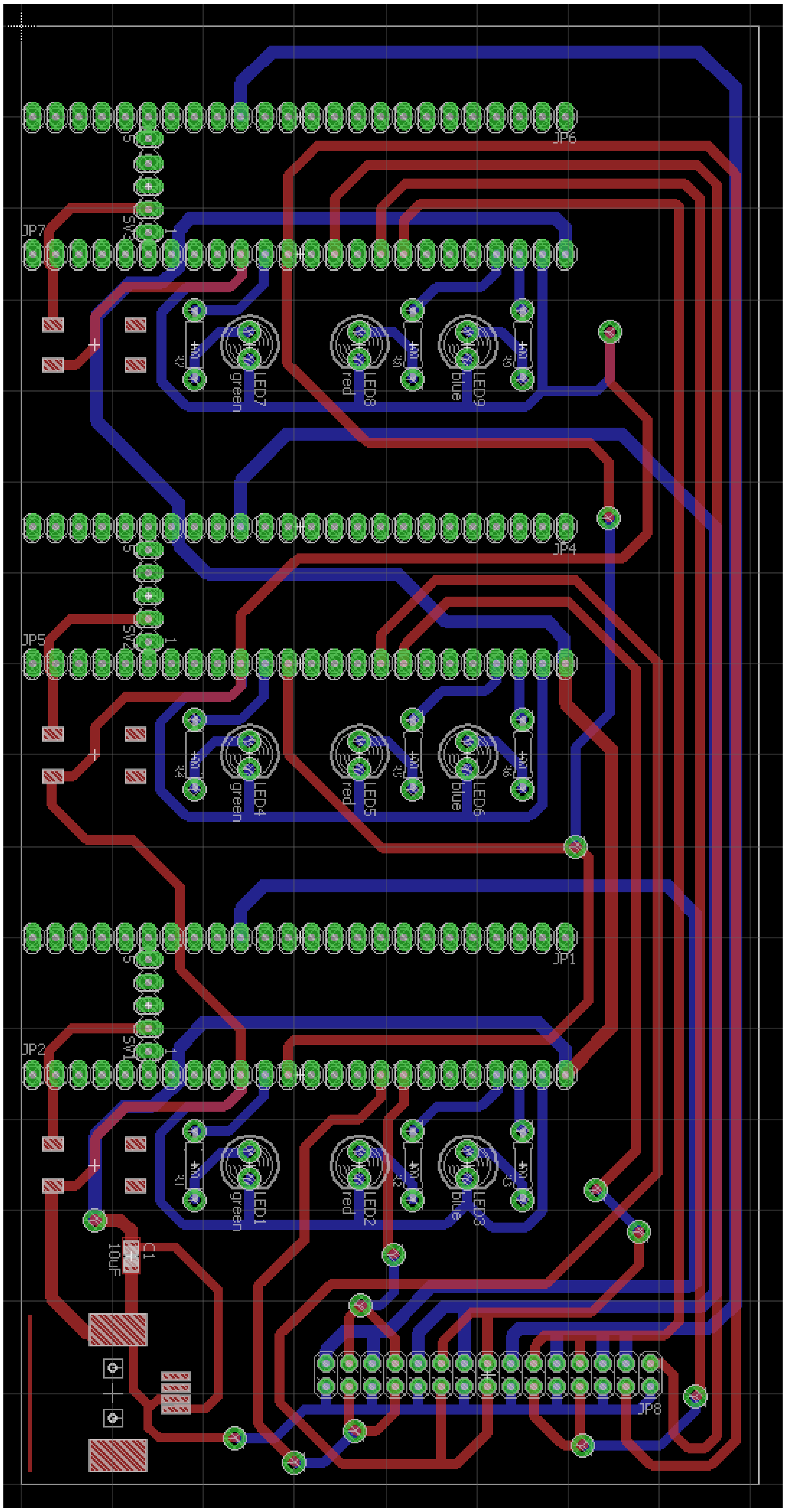


Figure A.6: The Layout for the Motherboard of Teensy 3.6 DAQ (16 cm × 8.1 cm with 1-cm grid on).

Appendix B

MATLAB Code

B.1 Data Acquisition MATLAB Code for QuickDAQ Software

```
clear, clc, close all

sample_rate = 15000;
num_sample = 7000;

ai0 = analoginput ('dtol', 0);
chan_1 = addchannel (ai0, 0);
chan_2 = addchannel (ai0, 1);
chan_3 = addchannel (ai0, 2);
chan_4 = addchannel (ai0, 3);
chan_5 = addchannel (ai0, 4);
chan_6 = addchannel (ai0, 5);
set (ai0, 'BufferingMode', 'Auto');
set (ai0, 'SampleRate', sample_rate);
set (ai0, 'SamplesPerTrigger', 15000);
set (ai0, 'TriggerType', 'Software');
set (ai0, 'TriggerChannel', chan_1);
set (ai0, 'TriggerCondition', 'Rising');
set (ai0, 'TriggerConditionValue', 0.8);
start(ai0);
[data, time] = getdata(ai0, num_sample);

t = time*1e3;
X_Eyeball = (-data(:,1) + 3 - 1.5 - 0.1)/0.174;
Y_Eyeball = (data(:,2) - 1.5 - 0.03)/0.174;
```

```
Z_Eyeball = (-data(:,3) + 3 - 1.5 - 0.11)/0.174;
X_Socket = (data(:,4) - 1.5 + 0.1)/0.174;
Y_Socket = (data(:,5) - 1.5 - 0.03)/0.174;
Z_Socket = (data(:,6) - 1.5 + 0.02)/0.174;

figure;
title('Impact on the front side of socket');
subplot(2, 3, 1);
plot(t, X_Eyeball, 'r-');
xlabel('Time (ms)');
ylabel('X output from sensor on eye ball (g)');
grid on;
subplot(2, 3, 2);
plot(t, Y_Eyeball, 'b-');
xlabel('Time (ms)');
ylabel('Y output from sensor on eye ball (g)');
grid on;
subplot(2, 3, 3);
plot(t, Z_Eyeball, 'g-');
xlabel('Time (ms)');
ylabel('Z output from sensor on eye ball (g)');
grid on;

subplot(2, 3, 4);
plot(t, X_Socket, 'r-');
xlabel('Time (ms)');
ylabel('X output from sensor on socket (g)');
grid on;
subplot(2, 3, 5);
plot(t, Y_Socket, 'b-');
xlabel('Time (ms)');
ylabel('Y output from sensor on socket (g)');
grid on;
```

```
subplot(2, 3, 6);
plot(t, Z_Socket, 'g-');
xlabel('Time (ms)');
ylabel('Z output from sensor on socket (g)');
grid on;
```

B.2 Data Processing MATLAB Code for Hit-by-Hammer Tests on Socket Model with Accelerometers

```
clear all
close all

fn='F1.xlsx';
[NUM,TXT,RAW] = xlsread(fn);

q = cell2mat(RAW(2:end,:));

% determine the impact start point.
% this is the first time when x-Eyeball (column 2) reached 0.174
% use the first 40000 samples to estimate the initial voltage offset.
% The nominal voltage offset is 1.5. The estimate differs from the nominal
% by about 2%. The impact start points determined by these two methods differ
    by about 2 samples (0.4 ms)
idx = find(abs(q(:,4)-mean(q(1:4000,4))) >= 0.174);
impactZeroIdx = idx(1);

% convert voltage to acceleration m/sec2
g = 9.80665;
q(:,2) = (-q(:,2)+3-1.5)/0.174*g; % column B, X-out-eyeball
q(:,3) = (q(:,3)-1.5)/0.174*g;    % column C, Y-out-eyeball
q(:,4) = (-q(:,4)+3-1.5)/0.174*g; % column D, Z-out-eyeball
q(:,5) = (q(:,5)-1.5)/0.174*g;    % column E, X-out-socket
q(:,6) = (q(:,6)-1.5)/0.174*g;    % column F, Y-out-socket
q(:,7) = (q(:,7)-1.5)/0.174*g;    % column G, Z-out-socket

t = q(:,1);
numSamples = size(q,1);
Fs = numSamples/(q(end,1)-q(1,1)); % sample frequency in Hz.
T = (q(end,1)-q(1,1))/numSamples;  % sample interval in s.
```

```
myTime = 0:T:q(end,1)-q(1,1);
myFreq = Fs*(0:(numSamples/2))/numSamples; % the frequency domain
impactZeroSec=t(impactZeroIdx); % time of the impact duration

% divide the data into before, impact and after segments
impactStartSec = 0.00; impactEndSec = 0.30;
[v, impactStartIdx]=min(abs(myTime - impactStartSec)); % determine impact
    starting time index.
[v, impactEndIdx]=min(abs(myTime - impactEndSec)); % determine impact ending
    time index.
before=q(1:impactStartIdx,2:end);
tBefore=myTime(1:impactStartIdx);
impact=q(impactStartIdx+1:impactEndIdx,2:end);
tImpact=myTime(impactStartIdx+1:impactEndIdx);
after=q(impactEndIdx+1:end,2:end);
tAfter=myTime(impactEndIdx+1:end);


% use the before data to determine instrument recording offset and noise
    frequency
offsets=mean(before); % different accel and different axis may have different
    offsets
% get data frequency properties
rmse = [];
noiseSpec = [];
impactSpec = [];
for i = 1:size(before,2)
    % correct the offsets so that initial accel is zero
    before(:,i) = before(:,i)-offsets(i);
    impact(:,i) = impact(:,i)-offsets(i);
    after(:,i) = after(:,i)-offsets(i);
    % compute system noise level
    rmse = [rmse, sqrt( sum( before(:,i).*before(:,i)) / impactStartIdx )];
```

```
    % because the offsets of the data have been corrected, the estimator is 0.
    % system noise frequency spectrum
    s = tBefore;
    X = before(:,i);
    [fBefore, P1, P2] = spec_1D_1_1(s,X);
    noiseSpec = [noiseSpec, P1];
    % impact frequency spectrum
    s = tImpact;
    X = impact(:,i);
    [fImpact, P1, P2] = spec_1D_1_1(s,X);
    impactSpec=[impactSpec, P1];
end

xEye = 1; % data columns designations
yEye = 2;
zEye = 3;
xSocket = 4;
ySocket = 5;
zSocket = 6;

dataNames = {'xEye','yEye','zEye','xSocket','ySocket','zSocket'};

% plot original data
subplot(6,3,1),plot(myTime,[before(:,xEye); impact(:,xEye); after(:,xEye)] );
title(['x-eyeball: offset = ' num2str(offsets(xEye)) '; rmse = ' num2str(rmse(
    xEye))]);
a = axis;
hold on
plot([tBefore(end), tBefore(end)],[a(3), a(4)],'r--');
plot([tImpact(end), tImpact(end)],[a(3), a(4)],'r--');
plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
hold off
ylabel('Acceleration (m/s^2)');
```

```
subplot(6,3,4),plot(myTime,[before(:,yEye); impact(:,yEye); after(:,yEye)]);
title(['y-eyeball: offset = ' num2str(offsets(yEye)) '; rmse = ' num2str(rmse(
    yEye))]);
a = axis;
hold on
plot([tBefore(end), tBefore(end)],[a(3), a(4)],'r--');
plot([tImpact(end), tImpact(end)],[a(3), a(4)],'r--');
plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
hold off
ylabel('Acceleration (m/s^2)');

subplot(6,3,7),plot(myTime,[before(:,zEye); impact(:,zEye); after(:,zEye)]);
title(['z-eyeball: offset = ' num2str(offsets(zEye)) '; rmse = ' num2str(rmse(
    zEye))]);
a = axis;
hold on
plot([tBefore(end), tBefore(end)],[a(3), a(4)],'r--');
plot([tImpact(end), tImpact(end)],[a(3), a(4)],'r--');
plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
hold off
ylabel('Acceleration (m/s^2)');

subplot(6,3,10),plot(myTime,[before(:,xSocket); impact(:,xSocket); after(:,
    xSocket)]);
title(['x-socket: offset = ' num2str(offsets(xSocket)) '; rmse = ' num2str(
    rmse(xSocket))]);
a = axis;
hold on
plot([tBefore(end), tBefore(end)],[a(3), a(4)],'r--');
plot([tImpact(end), tImpact(end)],[a(3), a(4)],'r--');
plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
hold off
```

```
ylabel('Acceleration (m/s^2)');

subplot(6,3,13),plot(myTime,[before(:,ySocket); impact(:,ySocket); after(:,
    ySocket)]);
title(['y-socket: offset = ' num2str(offsets(ySocket)) '; rmse = ' num2str(
    rmse(ySocket))]);
a = axis;
hold on
plot([tBefore(end), tBefore(end)],[a(3), a(4)],'r--');
plot([tImpact(end), tImpact(end)],[a(3), a(4)],'r--');
plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
hold off
ylabel('Acceleration (m/s^2)');

subplot(6,3,16),plot(myTime,[before(:,zSocket); impact(:,zSocket); after(:,
    zSocket)]);
title(['z-socket: offset = ' num2str(offsets(zSocket)) '; rmse = ' num2str(
    rmse(zSocket))]);
a = axis;
xlabel('Time (sec)');
hold on
plot([tBefore(end), tBefore(end)],[a(3), a(4)],'r--');
plot([tImpact(end), tImpact(end)],[a(3), a(4)],'r--');
plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
hold off
ylabel('Acceleration (m/s^2)');

% plot instrument spectrum
subplot(6,3,2),plot(fBefore,noiseSpec(:,xEye));
title(['x-eyeball: instrument spectrum ']);
a = axis;
subplot(6,3,5),plot(fBefore,noiseSpec(:,yEye));
title(['y-eyeball: instrument spectrum ']);
```

```
a = axis;
subplot(6,3,8),plot(fBefore,noiseSpec(:,zEye));
title(['z-eyeball: instrument spectrum ']);
a = axis;
subplot(6,3,11),plot(fBefore,noiseSpec(:,xSocket));
title(['x-socket: instrument spectrum ']);
a = axis;
subplot(6,3,14),plot(fBefore,noiseSpec(:,ySocket));
title(['y-socket: instrument spectrum ']);
a = axis;
subplot(6,3,17),plot(fBefore,noiseSpec(:,zSocket));
title(['z-socket: instrument spectrum ']);
a = axis;
xlabel('Frequency (Hz)');

% plot impact spectrum
subplot(6,3,3),plot(fImpact,impactSpec(:,xEye));
title(['x-eyeball: impact spectrum ']);
a = axis;
axis([a(1),500, a(3), a(4)]);
subplot(6,3,6),plot(fImpact,impactSpec(:,yEye));
title(['y-eyeball: impact spectrum ']);
a = axis;
axis([a(1),500, a(3), a(4)]);
subplot(6,3,9),plot(fImpact,impactSpec(:,zEye));
title(['z-eyeball: impact spectrum ']);
a = axis;
axis([a(1),500, a(3), a(4)]);
subplot(6,3,12),plot(fImpact,impactSpec(:,xSocket));
title(['x-socket: impact spectrum ']);
a=axis;
axis([a(1),500, a(3), a(4)]);
subplot(6,3,15),plot(fImpact,impactSpec(:,ySocket));
```

```
title(['y-socket: impact spectrum ']);
a=axis;
axis([a(1),500, a(3), a(4)]);
subplot(6,3,18),plot(fImpact,impactSpec(:,zSocket));
title(['z-socket: impact spectrum ']);
a=axis;
axis([a(1),500, a(3), a(4)]);
xlabel('Frequency (Hz)');
set(gcf,'name',[fn ': Converted Acceleration Data & Spectra before and during
    Impact'],'numbertitle','off')

% create a lowpass filter to remove noise
Fs=50000; % hz
lowpassHighCutoff = 500; % high freq cutoff in Hz
Wp = lowpassHighCutoff/(Fs/2); % pass band 0 to lowpassHighCutoff Hz
Ws = 1000/(Fs/2);  % stopband drop to 30 dB
[nL,WnL] = buttord(Wp,Ws,3,30);  % the order of the filter n and the cutoff
    frequency
% plot frequency response
[z,p,k] = butter(nL,WnL);
sos = zp2sos(z,p,k);
figure, freqz(sos,2048,Fs);
set(gcf,'name',[fn ': Lowpass Filter Frequency Response with high cutoff at '
    num2str(lowpassHighCutoff) ' Hz'],'numbertitle','off');

% create a highpass filter to remove slow drift
highpassLowCutoff=3; % low freq cut off in Hz
nH=3;
WnH=highpassLowCutoff/(Fs/2); % pass band above highpassLowCutoff Hz
[z,p,k] = butter(nH,WnH,'high');
% a third-order highpass Butterworth filter
% with a lowCutoff at highpassLowCutoff Hz.
sos = zp2sos(z,p,k);
```

```
figure, freqz(sos,2048,Fs);
set(gcf,'name',[fn ': Highpass Filter Frequency Response with low cutoff at '
    num2str(highpassLowCutoff) ' Hz'],'numbertitle','off');

% filter data
% it is difficult to make a bandpass filter with a very low low-freq cut
%and very high high-freq cut at one order
% Use one lowpass and one highpass filters instead.
% Without highpass filtering, the velocity and displacement would not
% settle to the initial condition when the impact is over.
[bL,aL] = butter(nL,WnL);
[bH,aH] = butter(nH,WnH,'high');
filtered = [];
figure;
for i=1:6
    data=impact(:,i); %[before(:,i); impact(:,i); after(:,i)];
    filtered = [filtered, filtfilt(bH, aH, filtfilt(bL,aL,data))];
    % Zero-phase forward and reverse digital IIR filtering.
    subplot(2,3,i),plot(tImpact, impact(:,i),'r');
    a=axis;
    hold on
    plot(tImpact,filtered(:,i),'b');
    plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
    hold off
    xlabel('Time (s)');
    ylabel('Acceleration (m/s^2)');
    title(dataNames{i});
end
set(gcf,'name',[fn ': Filtered Data during Impact'],'numbertitle','off')

% Get velocity
% use data within 0.3 sec after impact
[v, headCut]=min(abs(tImpact-impactZeroSec));
```

```
[v, tailCut]=min(abs(tImpact-0.3));
vel = [];
figure;
for i=1:6
    vel = [vel, cumtrapz(tImpact(headCut:tailCut),filtered(headCut:tailCut,i))
    ];
    subplot(2,3,i),plot(tImpact(headCut:tailCut), vel(:,i),'b');
    a=axis;
    hold on
    plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
    hold off
    xlabel('Time (s)');
    ylabel('Velocity (m/s)');
    title(dataNames{i});
end
set(gcf,'name',[fn ': Velocity'],'numbertitle','off')

% Get displacement
displace=[];
figure;
for i=1:6
    displace = [displace, cumtrapz(tImpact(headCut:tailCut),filtfilt(bH, aH,
    vel(:,i)))];  % the velocity is highpass filtered again
    subplot(2,3,i),plot(tImpact(headCut:tailCut), displace(:,i),'r');
    a=axis;
    hold on
    plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
    hold off
    xlabel('Time (s)');
    ylabel('Displacement (m)');
    title(dataNames{i});
end
set(gcf,'name',[fn ': Displacement'],'numbertitle','off')
```

```
% eye displacement in socket
figure;
plot(tImpact(headCut:tailCut), displace(:,4)-displace(:,1),'r');
a=axis;
title('Eye displacement in socket (xSocket - xEyeball)');
ylabel('Displacement (m)');
xlabel('Time (s)');
set(gcf,'name',[fn ': Eye Displacement in Socket'],'numbertitle','off')
```

B.3 MATLAB Function for Spectrum Analysis

```matlab
function [f, P1, P2]=spec_1D(s, X)
    L=numel(s);
    Fs=L/(s(end) - s(1)); % sample frequency in Hz
    T = round(1/Fs*L)/L;  % sample interval in sec. It is important to get round it up correctly
    t = 0 : T : (s(end) - s(1)); % the time sequence is in 0.00002 sec increments. The last element of myTime is 1.99998
    f = Fs*(0 : round(L/2))/L;

    Y = fft(X);
    P2 = abs(Y/L);
    P1 = P2(1 : round(L/2)+1);
    P1(2 : end - 1) = 2*P1(2 : end - 1);
end
```

B.4 Data Processing MATLAB Code for Rotation Tests on VOR Chair with Six IMU's

```
clear, clc, close all

%% Loading and preparing data

DATA_SD1 = 'SD_1/LOGGER17_Hit motion_12_CCW.csv';
DATA_SD2 = 'SD_2/LOGGER18_Hit motion_12_CCW.csv';
DATA_SD3 = 'SD_3/LOGGER19_Hit motion_12_CCW.csv';

gyroRange = 2000;                       % max deg/s
gyroSensitivity = 4.375;                % mdps/LSB for range of 125 deg/s
accelRange = 16;                        % max G force readable
accelSensitivity = 0.061;               % mg/LSB for range of 2 g
accelRange = bitsra(accelRange, 1);     % arithmetic right shift by 1 bit
impactDeg = 1.5;                        % degree threshold for impact
startTime = 2.5;
endTime = 5.2;

% read the testing data
[RAW_1] = csvread(DATA_SD1);
[RAW_2] = csvread(DATA_SD2);
[RAW_3] = csvread(DATA_SD3);
% merge the testing data
[RAW_Data] = [RAW_1(:, 1), RAW_1(:, 3:8), RAW_1(:, 10:15), RAW_2(:, 3:8),...
    RAW_2(:, 10:15), RAW_3(:, 3:8), RAW_3(:, 10:15)];

q_1 = RAW_Data(1 : end - 2, :);
temp = [RAW_1(1 : end - 2, 2), RAW_1(1 : end - 2, 9), RAW_2(1 : end - 2, 2),...
    RAW_2(1 : end - 2, 9), RAW_3(1 : end - 2, 2), RAW_3(1 : end - 2, 9)];

```

```
%%%%%%%%%% Calculate and convert the raw data to readable data

% Temperature compensations
% Raw temperature to degree C
temp = temp ./ 16 + ones(size(temp), 'like', temp) .* 25;

% Calculate sensitivities
dim_sensi = size(temp(:, 1));

accelSensi_PosteriorBrain = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 1) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_Forehead = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 2) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_MiddleBrain = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 3) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_LeftEye = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 4) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_FrontBrain = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 5) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_RightEye = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 6) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);

gyroSensi_PosteriorBrain = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 1) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_Forehead = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 2) - ones(dim_sensi(1, 1), 1)...
```

```
    .* 25) .* 0.015);
gyroSensi_MiddleBrain = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 3) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_LeftEye = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 4) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_FrontBrain = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 5) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_RightEye = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 6) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);

% Raw accelerations to m/s^2
g = 9.80665;

q_1(:, 5:7) = q_1(:, 5:7) .* (accelSensi_PosteriorBrain * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 11:13) = q_1(:, 11:13) .* (accelSensi_Forehead * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 17:19) = q_1(:, 17:19) .* (accelSensi_MiddleBrain * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 23:25) = q_1(:, 23:25) .* (accelSensi_LeftEye * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 29:31) = q_1(:, 29:31) .* (accelSensi_FrontBrain * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 35:37) = q_1(:, 35:37) .* (accelSensi_RightEye * ones(1, 3)...
    .* accelRange ./ 1000) .* g;

% Raw accelerations to g's
q_1(:, 5:7) = q_1(:, 5:7) ./ g;
q_1(:, 11:13) = q_1(:, 11:13) ./ g;
```

```
q_1(:, 17:19) = q_1(:, 17:19) ./ g;
q_1(:, 23:25) = q_1(:, 23:25) ./ g;
q_1(:, 29:31) = q_1(:, 29:31) ./ g;
q_1(:, 35:37) = q_1(:, 35:37) ./ g;

% Raw angular rates to deg/s
q_1(:, 2:4) = q_1(:, 2:4) .* (gyroSensi_PosteriorBrain * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 8:10) = q_1(:, 8:10) .* (gyroSensi_Forehead * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 14:16) = q_1(:, 14:16) .* (gyroSensi_MiddleBrain * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 20:22) = q_1(:, 20:22) .* (gyroSensi_LeftEye * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 26:28) = q_1(:, 26:28) .* (gyroSensi_FrontBrain * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 32:34) = q_1(:, 32:34) .* (gyroSensi_RightEye * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);

% Time stamp
t_stamp(1, 1) = 0;
Fs = 1/((q_1(end, 1) - q_1(1, 1))/(size(q_1, 1) - 1)/1000);
for i = 2 : size(q_1, 1)
    t_stamp(i, 1) = t_stamp(i - 1, 1) + (q_1(end, 1) - q_1(1, 1))/(size(q_1, 1) - 1);
end

% Merge two matrices, the resulting columns are
% Item                    Column #
% time                       1
% x_Forehead                 2
% y_Forehead                 3
% z_Forehead                 4
```

```
% x_LeftEye                5
% y_LeftEye                6
% z_LeftEye                7
% x_RightEye               8
% y_RightEye               9
% z_RightEye               10
% x_FrontBrain             11
% y_FrontBrain             12
% z_FrontBrain             13
% x_MiddleBrain            14
% y_MiddleBrain            15
% z_MiddleBrain            16
% x_PosteriorBrain         17
% y_PosteriorBrain         18
% z_PosteriorBrain         19
% pitch_Forehead           20
% yaw_Forehead             21
% roll_Forehead            22
% pitch_LeftEye            23
% yaw_LeftEye              24
% roll_LeftEye             25
% pitch_RightEye           26
% yaw_RightEye             27
% roll_RightEye            28
% pitch_FrontBrain         29
% yaw_FrontBrain           30
% roll_FrontBrain          31
% pitch_MiddleBrain        32
% yaw_MiddleBrain          33
% roll_MiddleBrain         34
% pitch_PosteriorBrain     35
% yaw_PosteriorBrain       36
% roll_PosteriorBrain      37
```

```
t_stamp = t_stamp ./ 1000; % convert ms to s

% Form the data matrix
q = [t_stamp, q_1(:, 11:13), q_1(:, 23:25), q_1(:, 35:37), q_1(:, 29:31),...
    q_1(:, 17:19), q_1(:, 5:7), q_1(:, 8:10), q_1(:, 20:22), q_1(:, 32:34),...
    q_1(:, 26:28), q_1(:, 14:16), q_1(:, 2:4)];

% use the before data to determine instrument recording offset and noise frequency
startIdx = 420;
before = q(1 : startIdx, 2 : end);
offsets = mean(before);

for i = 1 : 18
    q(:, i + 19) = q(:, i + 19) - offsets(i + 18);
end

%% Plot original data, instrument and impact spectra

x_Forehead = 1;
y_Forehead = 2;
z_Forehead = 3;
x_LeftEye = 4;
y_LeftEye = 5;
z_LeftEye = 6;
x_RightEye = 7;
y_RightEye = 8;
z_RightEye = 9;
x_FrontBrain = 10;
y_FrontBrain = 11;
z_FrontBrain = 12;
x_MiddleBrain = 13;
```

```
y_MiddleBrain = 14;
z_MiddleBrain = 15;
x_PosteriorBrain = 16;
y_PosteriorBrain = 17;
z_PosteriorBrain = 18;
pitch_Forehead = 19;
yaw_Forehead = 20;
roll_Forehead = 21;
pitch_LeftEye = 22;
yaw_LeftEye = 23;
roll_LeftEye = 24;
pitch_RightEye = 25;
yaw_RightEye = 26;
roll_RightEye = 27;
pitch_FrontBrain = 28;
yaw_FrontBrain = 29;
roll_FrontBrain = 30;
pitch_MiddleBrain = 31;
yaw_MiddleBrain = 32;
roll_MiddleBrain = 33;
pitch_PosteriorBrain = 34;
yaw_PosteriorBrain = 35;
roll_PosteriorBrain  = 36;

dataNames = {'Forehead', 'Left Eye', 'Right Eye', 'Front Brain',...
    'Middle Brain', 'Posterior Brain'};

%% Calculating angles from gyroscope data

% Pre-set the gain of the gyros (1 for now)
gyro_gain = 1;

% Create a matrix storing only calculated angles against time
```

```
angles = zeros(size(q, 1), 18);
impactIdx = zeros(1, 6);
impactTime = zeros(1, 6);
delayTime = zeros(1, 5);

% Get angles
for i = 1 : 18
    angles(:, i) = cumtrapz(q(:, 1), q(:, i + 19));
end

% Get impact times
for j = 1 : 6
    indx = find(abs(angles(:, 2 + (j - 1) * 3)) >= impactDeg);
    impactIdx(j) = indx(1);
    impactTime(j) = q(impactIdx(j), 1);
end

% Get delay times
for k = 1 : 5
    delayTime(k) = impactTime(k + 1) - impactTime(1);
end


%% Plot the angular rates

for p = 1 : 5
    figure('Name', sprintf('Angular Rate: %s vs Forehead',...
        dataNames{p + 1}));

    % Forehead IMU
    subplot(2, 3, 1), plot(q(:, 1), q(:, 20), 'r-.', 'LineWidth', 1.5);
    xlabel('Time (s)');
    ylabel('Degree/s');
```

```matlab
    title('Angular rates along x-axis of Forehead IMU');

    subplot(2, 3, 2), plot(q(:, 1), q(:, 21), 'b-.', 'LineWidth', 1.5);
    xlabel('Time (s)');
    ylabel('Degree/s');
    title('Angular rates along y-axis of Forehead IMU');

    subplot(2, 3, 3), plot(q(:, 1), q(:, 22), 'g-.', 'LineWidth', 1.5);
    xlabel('Time (s)');
    ylabel('Degree/s');
    title('Angular rates along z-axis of Forehead IMU');

    % Another IMU
    subplot(2, 3, 4), plot(q(:, 1), q(:, (p - 1) * 3 + 23), 'r',...
        'LineWidth', 1.5);
    hold on
    plot(q(:, 1), q(:, 20), 'k-.', 'LineWidth', 1.5);
    hold off
    xlabel('Time (s)');
    ylabel('Degree/s');
    title(sprintf('Angular rates along x-axis of %s IMU', dataNames{p + 1}));

    subplot(2, 3, 5), plot(q(:, 1), q(:, (p - 1) * 3 + 24), 'b',...
        'LineWidth', 1.5);
    hold on
    plot(q(:, 1), q(:, 21), 'k-.', 'LineWidth', 1.5);
    hold off
    xlabel('Time (s)');
    ylabel('Degree/s');
    title(sprintf('Angular rates along y-axis of %s IMU', dataNames{p + 1}));

    subplot(2, 3, 6), plot(q(:, 1), q(:, (p - 1) * 3 + 25), 'g',...
        'LineWidth', 1.5);
```

```
    hold on
    plot(q(:, 1), q(:, 22), 'k-.', 'LineWidth', 1.5);
    hold off
    xlabel('Time (s)');
    ylabel('Degree/s');
    title(sprintf('Angular rates along z-axis of %s IMU', dataNames{p + 1}));

end

%%%%%% All %%%%%%
figure('Name', 'All angular rates');

subplot(1, 3, 1), plot(q(:, 1), q(:, 23), 'r');
hold on
plot(q(:, 1), q(:, 26), 'b');
plot(q(:, 1), q(:, 29), 'r--');
plot(q(:, 1), q(:, 32), 'b--');
plot(q(:, 1), q(:, 35), 'g--');
plot(q(:, 1), q(:, 20), 'k--');
hold off
legend('Left Eye', 'Right Eye', 'Front Brain', 'Middle Brain',...
    'Posterior Brain', 'Forehead');
xlabel('Time (s)');
ylabel('Degree/s');
title('Angular rates along x-axis of the IMUs)');
subplot(1, 3, 2), plot(q(:, 1), q(:, 24), 'r');
hold on
plot(q(:, 1), q(:, 27), 'b');
plot(q(:, 1), q(:, 30), 'r--');
plot(q(:, 1), q(:, 33), 'b--');
plot(q(:, 1), q(:, 36), 'g--');
plot(q(:, 1), q(:, 21), 'k--');
hold off
```

```
legend('Left Eye', 'Right Eye', 'Front Brain', 'Middle Brain',...
    'Posterior Brain', 'Forehead');
xlabel('Time (s)');
ylabel('Degree/s');
title('Angular rates along y-axis of the IMUs');
subplot(1, 3, 3), plot(q(:, 1), q(:, 25), 'r');
hold on
plot(q(:, 1), q(:, 28), 'b');
plot(q(:, 1), q(:, 31), 'r--');
plot(q(:, 1), q(:, 34), 'b--');
plot(q(:, 1), q(:, 37), 'g--');
plot(q(:, 1), q(:, 22), 'k--');
hold off
legend('Left Eye', 'Right Eye', 'Front Brain', 'Middle Brain',...
    'Posterior Brain', 'Forehead');
xlabel('Time (s)');
ylabel('Degree/s');
title('Angular rates along z-axis of the IMUs');

%% Plot the angles

for p = 1 : 5
    figure('Name', sprintf('Integrated Angle: %s vs Forehead',...
        dataNames{p + 1}));

    % Forehead IMU
    subplot(2, 3, 1), plot(q(:, 1), angles(:, 1), 'r-.', 'LineWidth', 1.5);
    xlabel('Time (s)');
    ylabel('Angle (deg)');
    title('Integrated angles along x-axis of Forehead IMU');

    subplot(2, 3, 2), plot(q(:, 1), angles(:, 2), 'b-.', 'LineWidth', 1.5);
    xlabel('Time (s)');
```

```
    ylabel('Angle (deg)');
    title('Angles along y-axis of Forehead IMU');

    subplot(2, 3, 3), plot(q(:, 1), angles(:, 3), 'g-.', 'LineWidth', 1.5);
    xlabel('Time (s)');
    ylabel('Angle (deg)');
    title('Integrated angles along z-axis of Forehead IMU');

    % Another IMU
    subplot(2, 3, 4), plot(q(:, 1), angles(:, (p - 1) * 3 + 4), 'r',...
        'LineWidth', 1.5);
    hold on
    plot(q(:, 1), angles(:, 1), 'k-.', 'LineWidth', 1.5);
    hold off
    xlabel('Time (s)');
    ylabel('Angle (deg)');
    title(sprintf('Integrated angles along x-axis of %s IMU', dataNames{p + 1}));

    subplot(2, 3, 5), plot(q(:, 1), angles(:, (p - 1) * 3 + 5), 'b',...
        'LineWidth', 1.5);
    hold on
    plot(q(:, 1), angles(:, 2), 'k-.', 'LineWidth', 1.5);
    hold off
    xlabel('Time (s)');
    ylabel('Angle (deg)');
    title(sprintf('Integrated angles along y-axis of %s IMU', dataNames{p + 1}));

    subplot(2, 3, 6), plot(q(:, 1), angles(:, (p - 1) * 3 + 6), 'g',...
        'LineWidth', 1.5);
    hold on
    plot(q(:, 1), angles(:, 3), 'k-.', 'LineWidth', 1.5);
```

```
    hold off
    xlabel('Time (s)');
    ylabel('Angle (deg)');
    title(sprintf('Integrated angles along z-axis of %s IMU', dataNames{p + 1}));

end

%%%%%% Map the angles to the global rotation %%%%%%
% Hard-coded here for the interested time range
newStart = round(startTime*Fs);
newEnd = round(endTime*Fs);
newTime = q(newStart : newEnd, 1);
newAngles = map(angles(newStart : newEnd, :), 0, 160);

%%% Calculate new delay times %%%
impactIdxNew = zeros(1, 6);
impactTimeNew = zeros(1, 6);
delayTimeNew = zeros(1, 5);

% Get impact times
for j = 1 : 6
    indxNew = find(abs(newAngles(:, 2 + (j - 1) * 3)) >= impactDeg);
    impactIdxNew(j) = indxNew(1);
    impactTimeNew(j) = q(impactIdxNew(j), 1);
end

% Get delay times
for k = 1 : 5
    delayTimeNew(k) = impactTimeNew(k + 1) - impactTimeNew(1);
end

%%% Only y-axis %%%
```

```

figure('Name', 'Integrated angles along y-axis at rotation start');
plot(newTime, newAngles(:, 5), 'r', 'LineWidth', 1.2);
a = axis;
hold on
plot(newTime, newAngles(:, 8), 'b', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 11), 'r-.', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 14), 'b-.', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 17), 'g-.', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 2), 'k--', 'LineWidth', 1.2);
plot([a(1), a(2)],[1.5, 1.5], 'b--', 'LineWidth', 1.2);
plot([a(1), a(2)],[1, 1], 'g--', 'LineWidth', 1.2);
plot([a(1), a(2)],[0.5, 0.5], 'r--', 'LineWidth', 1.2);
hold off
grid on
legend('Left Eye', 'Right Eye', 'Front Brain', 'Middle Brain',...
    'Posterior Brain', 'Forehead', 'Threshold of 1.5 degree',...
    'Threshold of 1 degree', 'Threshold of 0.5 degree', 'Location',...
    'northwest');
xlabel('Time (s)', 'FontSize', 12);
ylabel('Angle (deg)', 'FontSize', 12);
title('Integrated angles along y-axis of the IMUs (theta) at rotation start'
    ,...
    'FontSize', 12);

figure('Name', 'Integrated angles along y-axis at rotation end');
plot(newTime, newAngles(:, 5), 'r', 'LineWidth', 1.2);
a = axis;
hold on
plot(newTime, newAngles(:, 8), 'b', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 11), 'r-.', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 14), 'b-.', 'LineWidth', 1.2);
plot(newTime, newAngles(:, 17), 'g-.', 'LineWidth', 1.2);
```

```
plot(newTime, newAngles(:, 2), 'k--', 'LineWidth', 1.2);
plot([a(1), a(2)],[159.5, 159.5], 'r--', 'LineWidth', 1.2);
plot([a(1), a(2)],[159, 159], 'g--', 'LineWidth', 1.2);
plot([a(1), a(2)],[158.5, 158.5], 'b--', 'LineWidth', 1.2);
hold off
grid on
legend('Left Eye', 'Right Eye', 'Front Brain', 'Middle Brain',...
    'Posterior Brain', 'Forehead', 'Threshold of 159.5 degree',...
    'Threshold of 159 degree', 'Threshold of 158.5 degree', 'Location',...
    'southeast');
xlabel('Time (s)', 'FontSize', 12);
ylabel('Angle (deg)', 'FontSize', 12);
title('Integrated angles along y-axis of the IMUs (theta) at rotation end',...
    'FontSize', 12);
```

B.5 Data Processing MATLAB Code for Drop-and-Impact Tests on Skull Model with Six IMU's

```matlab
%% Loading and preparing data
DATA_SD1 = 'SD_1/LOGGER03.csv';
DATA_SD2 = 'SD_2/LOGGER03.csv';
DATA_SD3 = 'SD_3/LOGGER03.csv';

gyroRange = 2000;            % Max deg/s
gyroSensitivity = 4.375;     % mdps/LSB for range of 125 deg/s
accelRange = 16;             % Max G force readable
accelSensitivity = 0.061;    % mg/LSB for range of 2 g
accelRange = bitsra(accelRange, 1); % Arithmetic right shift by 1 bit

% read the testing data
[RAW_1] = csvread(DATA_SD1);
[RAW_2] = csvread(DATA_SD2);
[RAW_3] = csvread(DATA_SD3);
% merge the testing data
[RAW_Data] = [RAW_1(:, 1), RAW_1(:, 3:8), RAW_1(:, 10:15), RAW_2(:, 3:8),...
    RAW_2(:, 10:15), RAW_3(:, 3:8), RAW_3(:, 10:15)];

% for sinusoidal data
q_1 = RAW_Data(1 : end - 2, :);
temp = [RAW_1(1 : end - 2, 2), RAW_1(1 : end - 2, 9), RAW_2(1 : end - 2, 2)
    ,...
    RAW_2(1 : end - 2, 9), RAW_3(1 : end - 2, 2), RAW_3(1 : end - 2, 9)];

%%%%%%%%%% Calculate and convert the raw data to readable data

% Temperature compensations
% Raw temperature to degree C
temp = temp ./ 16 + ones(size(temp), 'like', temp) .* 25;
```

```
% Calculate sensitivities
dim_sensi = size(temp(:, 1));

accelSensi_Skull = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 1) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_RightEye = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 2) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_LeftEye = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 3) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_FrontBrain = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 4) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_MiddleBrain = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 5) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);
accelSensi_PosteriorBrain = accelSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 6) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.01);

gyroSensi_Skull = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 1) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_RightEye = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 2) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_LeftEye = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 3) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_FrontBrain = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
```

```
    (ones(dim_sensi(1, 1), 1) + (temp(:, 4) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_MiddleBrain = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 5) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);
gyroSensi_PosteriorBrain = gyroSensitivity .* ones(dim_sensi(1, 1), 1) .*...
    (ones(dim_sensi(1, 1), 1) + (temp(:, 6) - ones(dim_sensi(1, 1), 1)...
    .* 25) .* 0.015);

% Raw accelerations to m/s^2
g = 9.80665;

q_1(:, 5:7) = q_1(:, 5:7) .* (accelSensi_Skull * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 11:13) = q_1(:, 11:13) .* (accelSensi_RightEye * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 17:19) = q_1(:, 17:19) .* (accelSensi_LeftEye * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 23:25) = q_1(:, 23:25) .* (accelSensi_FrontBrain * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 29:31) = q_1(:, 29:31) .* (accelSensi_MiddleBrain * ones(1, 3)...
    .* accelRange ./ 1000) .* g;
q_1(:, 35:37) = q_1(:, 35:37) .* (accelSensi_PosteriorBrain * ones(1, 3)...
    .* accelRange ./ 1000) .* g;

% Raw angular rates to deg/s
q_1(:, 2:4) = q_1(:, 2:4) .* (gyroSensi_Skull * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 8:10) = q_1(:, 8:10) .* (gyroSensi_RightEye * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 14:16) = q_1(:, 14:16) .* (gyroSensi_LeftEye * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 20:22) = q_1(:, 20:22) .* (gyroSensi_FrontBrain * ones(1, 3)...
```

```
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 26:28) = q_1(:, 26:28) .* (gyroSensi_MiddleBrain * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);
q_1(:, 32:34) = q_1(:, 32:34) .* (gyroSensi_PosteriorBrain * ones(1, 3)...
    .* gyroRange ./ 125 ./ 1000);

% Time stamp
t_stamp(1, 1) = 0;
for i = 2 : size(q_1, 1)
    t_stamp(i, 1) = t_stamp(i - 1, 1) + (q_1(end, 1) - q_1(1, 1))/(size(q_1, 1) - 1);
end

t_stamp = t_stamp ./ 1000; % convert ms to s

% Form the data matrix
q = [t_stamp, q_1(:, 5:7), q_1(:, 11:13), q_1(:, 17:19),...
    - q_1(:, 23:24), q_1(:, 25), - q_1(:, 29:30), q_1(:, 31),...
    - q_1(:, 35:36), q_1(:, 37),...
    q_1(:, 2:4), q_1(:, 8:10), q_1(:, 14:16),...
    - q_1(:, 20:21), q_1(:, 22), - q_1(:, 26:27), q_1(:, 28),...
    - q_1(:, 32:33), q_1(:, 34)];

%% Calibraiton of the gyroscope

instr_trace_back = 300;

startIdx = 800;
% stopIdx = 5600;
before = q(1 : startIdx, 2 : end);
offsets = mean(before);

for i = 1 : 18
```

```
    q(:, i + 19) = q(:, i + 19) - offsets(i + 18);
end

t = q(:, 1);
numSamples = size(q, 1);
Fs = numSamples/(q(end, 1) - q(1, 1)); % sample frequency in Hz.
dt = (q(end, 1) - q(1, 1))/numSamples;  % sample interval in sec.
delay_2 = round(0.0094 / dt);
delay_3 = round(0.0133 / dt); %%% for #03

for i = numSamples : -1 : delay_2

    q(i, 8 : 13) = q(i - delay_2 + 1, 8 : 13);
    q(i, 26 : 31) = q(i - delay_2 + 1, 26 : 31);

end

for j = numSamples : -1 : delay_3

    q(j, 14 : 19) = q(j - delay_3 + 1, 14 : 19);
    q(j, 32 : 37) = q(j - delay_3 + 1, 32 : 37);

end


%% Divide the time

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Note: The time division boundaries have   %%%
%%% to be changed for different data sections.%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fallingIdx = 1050;

```

```
%%%%%%%%%%% Determine the impact start point. [with trigger]
idx_2 = find(q(fallingIdx : end, 4) >= 5 * g);
impactIdx = fallingIdx + idx_2(2) - 5;

impactZeroSec = t(impactIdx); % time of the impact duration

% Divide the data into falling, impact and after segments
fallingStartSec = t(fallingIdx); impactStartSec = t(impactIdx);
impactEndIdx0 = round(0.777 / dt);
impactEndSec = t(impactEndIdx0);

[v, fallingStartIdx] = min(abs(t - fallingStartSec)); % determine falling starting time index.
[v, impactEndIdx] = min(abs(t - impactEndSec)); % determine impact ending time index.
before = q(1 : fallingStartIdx, 2 : 37); % Notice: matrix without time stamp!
tBefore = t(1 : fallingStartIdx); % Notice: pure time stamp.
falling = q(fallingStartIdx + 1 : impactIdx, 2 : 37);
tFalling = t(fallingStartIdx + 1 : impactIdx);
impact = q(impactIdx + 1 : impactEndIdx, 2 : 37);
tImpact = t(impactIdx + 1 : impactEndIdx);
after = q(impactEndIdx + 1 : end, 2 : 37);
tAfter = t(impactEndIdx + 1 : end);


%% Plot original data, instrument and impact spectra

% get data frequency properties
rmse = [];
noiseSpec = [];
impactSpec = [];
for i = 1 : size(before, 2)
    % compute system noise level
```

```
    rmse = [rmse, sqrt(sum(before(:,i).*before(:,i))/impactIdx)];
    % system noise frequency spectrum
    s = tBefore;
    X = before(1 : fallingIdx - instr_trace_back, i);
    [fBefore, P1, P2] = spec_1D(s, X);
    noiseSpec = [noiseSpec, P1];
    % impact frequency spectrum
    s = tImpact;
    X = impact(:, i);
    [fImpact, P1, P2] = spec_1D(s, X);
    impactSpec = [impactSpec, P1];
end

xSkull_accel = 1; % data columns designations
ySkull_accel = 2;
zSkull_accel = 3;
xRightEye_accel = 4;
yRightEye_accel = 5;
zRightEye_accel = 6;
xLeftEye_accel = 7;
yLeftEye_accel = 8;
zLeftEye_accel = 9;
xFrontBrain_accel = 10;
yFrontBrain_accel = 11;
zFrontBrain_accel = 12;
xMiddleBrain_accel = 13;
yMiddleBrain_accel = 14;
zMiddleBrain_accel = 15;
xPosteriorBrain_accel = 16;
yPosteriorBrain_accel = 17;
zPosteriorBrain_accel = 18;
xSkull_gyro = 19;
ySkull_gyro = 20;
```

```
zSkull_gyro = 21;
xRightEye_gyro = 22;
yRightEye_gyro = 23;
zRightEye_gyro = 24;
xLeftEye_gyro = 25;
yLeftEye_gyro = 26;
zLeftEye_gyro = 27;
xFrontBrainEye_gyro = 28;
yFrontBrainEye_gyro = 29;
zFrontBrainEye_gyro = 30;
xMiddleBrainEye_gyro = 31;
yMiddleBrainEye_gyro = 32;
zMiddleBrainEye_gyro = 33;
xPosteriorBrainEye_gyro = 34;
yPosteriorBrainEye_gyro = 35;
zPosteriorBrainEye_gyro = 36;
% trigger = 37;

dataNames = {'xSkullAccel', 'ySkullAccel', 'zSkullAccel',...
    'xRightEyeAccel', 'yRightEyeAccel', 'zRightEyeAccel',...
    'xLeftEyeAccel', 'yLeftEyeAccel', 'zLeftEyeAccel',...
    'xFrontBrainAccel', 'yFrontBrainAccel', 'zFrontBrainAccel',...
    'xMiddleBrainAccel', 'yMiddleBrainAccel', 'zMiddleBrainAccel',...
    'xPosteriorBrainAccel', 'yPosteriorBrainAccel', 'zPosteriorBrainAccel',...
    'xSkullGyro', 'ySkullGyro', 'zSkullGyro',...
    'xRightEyeGyro', 'yRightEyeGyro', 'zRightEyeGyro',...
    'xLeftEyeGyro', 'yLeftEyeGyro', 'zLeftEyeGyro',...
    'xFrontBrainGyro', 'yFrontBrainGyro', 'zFrontBrainGyro',...
    'xMiddleBrainGyro', 'yMiddleBrainGyro', 'zMiddleBrainGyro',...
    'xPosteriorBrainGyro', 'yPosteriorBrainGyro', 'zPosteriorBrainGyro',...
    'trigger'};

```

```
IMUNames = {'Skull', 'Right Eye', 'Left Eye', 'Front Brain', 'Middle Brain',...
    'Posterior Brain'};

%%%%%%%%%% plot original accelerometer data
for m = 1 : 5

    figure('Name', sprintf('Converted Accel Data and Spectra: %s vs Skull', IMUNames{m + 1}));

    for n = 1 : 3

        subplot(6, 3, 3*n - 2), plot(t, [before(:, 3*m + n); falling(:, 3*m + n); impact(:, 3*m + n); after(:, 3*m + n)]);
        title([dataNames{3*m + n} ': offset = ' num2str(offsets(3*m + n)) '; rmse = ' num2str(rmse(3*m + n))]);
        a = axis;
        hold on
        plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.');
        plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--');
        plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g');
        hold off
        xlabel('Time (s)');
        ylabel('Acceleration (m/s^2)');

        subplot(6, 3, 3*n + 7), plot(t, [before(:, n); falling(:, n); impact(:, n); after(:, n)]);
        title([dataNames{n} ': offset = ' num2str(offsets(n)) '; rmse = ' num2str(rmse(n))]);
        a = axis;
        hold on
        plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.');
        plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--');
```

```
        plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g');
        hold off
        xlabel('Time (s)');
        ylabel('Acceleration (m/s^2)');

        % plot instrument spectrum

        subplot(6, 3, 3*n - 1), plot(fBefore, noiseSpec(:, 3*m + n));
        title([dataNames{3*m + n} ': instrument spectrum']);
        a = axis;
        xlabel('Frequency (Hz)');

        subplot(6, 3, 3*n + 8),plot(fBefore, noiseSpec(:, n));
        title([dataNames{n} ': instrument spectrum'] );
        a = axis;
        xlabel('Frequency (Hz)');

        % plot impact spectrum

        subplot(6, 3, 3*n), plot(fImpact, impactSpec(:, 3*m + n));
        title([dataNames{3*m + n} ': impact spectrum']);
        a = axis;
        axis([a(1),500, a(3), a(4)]);
        xlabel('Frequency (Hz)');

        subplot(6, 3, 3*n + 9), plot(fImpact, impactSpec(:, n));
        title([dataNames{n} ': impact spectrum']);
        a=axis;
        axis([a(1),500, a(3), a(4)]);
        xlabel('Frequency (Hz)');

    end
end
```

```
%%%%%%%%%% plot original gyroscope data
for m = 1 : 5

    figure('Name', sprintf('Converted Gyro Data and Spectra: %s vs Skull', IMUNames{m + 1}));

    for n = 1 : 3

        subplot(6, 3, 3*n - 2), plot(t, [before(:, 3*m + n + 18); falling(:, 3*m + n + 18); impact(:, 3*m + n + 18); after(:, 3*m + n + 18)]);
        title([dataNames{3*m + n + 18} ': offset = ' num2str(offsets(3*m + n + 18)) '; rmse = ' num2str(rmse(3*m + n + 18))]);
        a = axis;
        hold on
        plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.');
        plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--');
        plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g');
        hold off
        xlabel('Time (s)');
        ylabel('Angular Rate (o/s)');

        subplot(6, 3, 3*n + 7), plot(t, [before(:, n + 18); falling(:, n + 18); impact(:, n + 18); after(:, n + 18)]);
        title([dataNames{n + 18} ': offset = ' num2str(offsets(n + 18)) '; rmse = ' num2str(rmse(n + 18))]);
        a = axis;
        hold on
        plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.');
        plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--');
        plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g');
        hold off
        xlabel('Time (s)');
```

```
        ylabel('Angular Rate (o/s)');

        % plot instrument spectrum

        subplot(6, 3, 3*n - 1), plot(fBefore, noiseSpec(:, 3*m + n + 18));
        title([dataNames{3*m + n + 18} ': instrument spectrum']);
        a = axis;
        xlabel('Frequency (Hz)');

        subplot(6, 3, 3*n + 8),plot(fBefore, noiseSpec(:, n + 18));
        title([dataNames{n + 18} ': instrument spectrum'] );
        a = axis;
        xlabel('Frequency (Hz)');

        % plot impact spectrum

        subplot(6, 3, 3*n), plot(fImpact, impactSpec(:, 3*m + n + 18));
        title([dataNames{3*m + n + 18} ': impact spectrum']);
        a = axis;
        axis([a(1),500, a(3), a(4)]);
        xlabel('Frequency (Hz)');

        subplot(6, 3, 3*n + 9), plot(fImpact, impactSpec(:, n + 18));
        title([dataNames{n + 18} ': impact spectrum']);
        a=axis;
        axis([a(1),500, a(3), a(4)]);
        xlabel('Frequency (Hz)');

    end
end

%% LPF the accelerations

```

```
%%%%%%%%%% Create a lowpass filter to remove accelerometer's noise
lowpassCutoff = 140; % high freq cutoff in Hz
Wp = lowpassCutoff/(Fs/2); % pass band 0 to lowpassHighCutoff Hz
Ws = 500/(Fs/2);  % stopband drop to 30 dB
[nL, WnL] = buttord(Wp, Ws, 3, 30);  % the order of the filter n and the cutoff frequency
% plot frequency response
[z, p, k] = butter(nL, WnL);
sos = zp2sos(z, p, k);
figure, freqz(sos, 2048, Fs);
set(gcf, 'name', [DATA_SD1 ': Lowpass Filter Frequency Response with cutoff at ' num2str(lowpassCutoff) ' Hz'],'numbertitle','off');

%%%%%%%%%% Filter the raw data
[bL, aL] = butter(nL, WnL);
filtered_accel = [];
data = [before; falling; impact; after];
t_data_1 = [tBefore; tFalling; tImpact; tAfter];

% Zero-phase forward and reverse digital IIR filtering.

figure;
for i = 1 : 6
    data_Z_accel = data(:, 3*i);
    filtered_accel = [filtered_accel, filtfilt(bL, aL, data_Z_accel)];
    % Zero-phase forward and reverse digital IIR filtering.
    subplot(2, 3, i), plot(t_data_1, data(:, 3*i), 'r');
    a = axis;
    hold on
    plot(t_data_1, filtered_accel(:, i), 'b');
    plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.');
    plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--');
    plot([impactZeroSec,impactZeroSec],[a(3), a(4)],'g');
```

```
    hold off
    xlabel('Time (s)');
    ylabel('Acceleration (m/s^2)');
    str0 = sprintf('Filtered %s data with LPFcut = %f Hz', dataNames{3*i}, lowpassCutoff);
    title(str0);
end
set(gcf,'name',[DATA_SD1 ': Filtered Accel Data during Fall and Impact'],'numbertitle','off')


%% Calculating angles from accelerometer data

% Calculate the tilt angles between each sensor's coordinate system and the
% universal coordinate system (in which gravity is in the -z direction).
% Theta for the angle between the x-axis and universal X-Y plane, phi for the angle
% between the y-axis and universal X-Y plane, psi for the angle between
% z-axis and universal +Z-axis.
X_actual_skull = q(:, 2);
Y_actual_skull = q(:, 3);
Z_actual_skull = q(:, 4);
X_actual_rightEye = q(:, 5);
Y_actual_rightEye = q(:, 6);
Z_actual_rightEye = q(:, 7);
X_actual_leftEye = q(:, 8);
Y_actual_leftEye = q(:, 9);
Z_actual_leftEye = q(:, 10);
X_actual_frontBrain = q(:, 11);
Y_actual_frontBrain = q(:, 12);
Z_actual_frontBrain = q(:, 13);
X_actual_middleBrain = q(:, 14);
Y_actual_middleBrain = q(:, 15);
```

```
Z_actual_middleBrain = q(:, 16);
X_actual_posteriorBrain = q(:, 17);
Y_actual_posteriorBrain = q(:, 18);
Z_actual_posteriorBrain = q(:, 19);

theta_skull = atan2d(X_actual_skull, sqrt(Y_actual_skull.^2 + Z_actual_skull.^2)); % [degrees]
phi_skull = atan2d(Y_actual_skull, sqrt(X_actual_skull.^2 + Z_actual_skull.^2));
psi_skull = atan2d(sqrt(X_actual_skull.^2 + Y_actual_skull.^2), Z_actual_skull);

theta_rightEye = atan2d(X_actual_rightEye, sqrt(Y_actual_rightEye.^2 + Z_actual_rightEye.^2));
phi_rightEye = atan2d(Y_actual_rightEye, sqrt(X_actual_rightEye.^2 + Z_actual_rightEye.^2));
psi_rightEye = atan2d(sqrt(X_actual_rightEye.^2 + Y_actual_rightEye.^2), Z_actual_rightEye);

theta_leftEye = atan2d(X_actual_leftEye, sqrt(Y_actual_leftEye.^2 + Z_actual_leftEye.^2));
phi_leftEye = atan2d(Y_actual_leftEye, sqrt(X_actual_leftEye.^2 + Z_actual_leftEye.^2));
psi_leftEye = atan2d(sqrt(X_actual_leftEye.^2 + Y_actual_leftEye.^2), Z_actual_leftEye);

theta_frontBrain = atan2d(X_actual_frontBrain, sqrt(Y_actual_frontBrain.^2 + Z_actual_frontBrain.^2));
phi_frontBrain = atan2d(Y_actual_frontBrain, sqrt(X_actual_frontBrain.^2 + Z_actual_frontBrain.^2));
psi_frontBrain = atan2d(sqrt(X_actual_frontBrain.^2 + Y_actual_frontBrain.^2), Z_actual_frontBrain);

```

```
theta_middleBrain = atan2d(X_actual_middleBrain, sqrt(Y_actual_middleBrain.^2 + Z_actual_middleBrain.^2));
phi_middleBrain = atan2d(Y_actual_middleBrain, sqrt(X_actual_middleBrain.^2 + Z_actual_middleBrain.^2));
psi_middleBrain = atan2d(sqrt(X_actual_middleBrain.^2 + Y_actual_middleBrain.^2), Z_actual_middleBrain);

theta_posteriorBrain = atan2d(X_actual_posteriorBrain, sqrt(Y_actual_posteriorBrain.^2 + Z_actual_posteriorBrain.^2));
phi_posteriorBrain = atan2d(Y_actual_posteriorBrain, sqrt(X_actual_posteriorBrain.^2 + Z_actual_posteriorBrain.^2));
psi_posteriorBrain = atan2d(sqrt(X_actual_posteriorBrain.^2 + Y_actual_posteriorBrain.^2), Z_actual_posteriorBrain);

psi = [psi_skull, psi_rightEye, psi_leftEye, psi_frontBrain, psi_middleBrain, psi_posteriorBrain];

%% use only the acceleration in z axes of the accels to compute the tilt
%% angles to gravity ------ will be further used to compensate tilt angles
%% given by the gyros
% eye_tilt_accel = asind(Y_actual_eye(1 : fallingStartIdx - 100, 1) ./ g);
% head_tilt_accel = asind(Y_actual_head(1 : fallingStartIdx - 100, 1) ./ g);

% Project the actual accelerations in each axis of each sensor used to the
% universal coordinate system. We have already set the Z axis along the
% gravity yet we need to assume the X (or Y) axis to set the universal
% coordinate system. Since the +x of the sensor on the forehead was tilted
% the lest, we assume X_actual_head as the +X of the universal coordinate
% system. Then phi_head is the angle between y_head and Y_universal,
% psi_head is the angle between z-head and Z_universal.
Z_skull_accel = (Z_actual_skull.*cosd(psi_skull) - Y_actual_skull.*sind(phi_skull)...
    - X_actual_skull.*sind(phi_skull)) ./ 3;
```

```
Z_rightEye_accel = (Z_actual_rightEye.*cosd(psi_rightEye) - Y_actual_rightEye
    .*sind(phi_rightEye)...
    - X_actual_rightEye.*sind(phi_rightEye)) ./ 3;
Z_leftEye_accel = (Z_actual_leftEye.*cosd(psi_leftEye) - Y_actual_leftEye.*
    sind(phi_leftEye)...
    - X_actual_leftEye.*sind(phi_leftEye)) ./ 3;
Z_frontBraine_accel = (Z_actual_frontBrain.*cosd(psi_frontBrain) -
    Y_actual_frontBrain.*sind(phi_frontBrain)...
    - X_actual_frontBrain.*sind(phi_frontBrain)) ./ 3;
Z_middleBraine_accel = (Z_actual_middleBrain.*cosd(psi_middleBrain) -
    Y_actual_middleBrain.*sind(phi_middleBrain)...
    - X_actual_middleBrain.*sind(phi_middleBrain)) ./ 3;
Z_posteriorBraine_accel = (Z_actual_posteriorBrain.*cosd(psi_posteriorBrain) -
     Y_actual_posteriorBrain.*sind(phi_posteriorBrain)...
    - X_actual_posteriorBrain.*sind(phi_posteriorBrain)) ./ 3;

Ur_rightEye_accel =  Z_rightEye_accel - Z_skull_accel;
Ur_leftEye_accel =  Z_leftEye_accel - Z_skull_accel;
Ur_frontBraine_accel =  Z_frontBraine_accel - Z_skull_accel;
Ur_middleBraine_accel =  Z_middleBraine_accel - Z_skull_accel;
Ur_posteriorBraine_accel =  Z_posteriorBraine_accel - Z_skull_accel;


%% Calculating angles from gyroscope data

% Pre-set the gain of the gyros (1 for now)
gyro_gain = 1;

% Create a matrix storing only calculated angles against time
angles = zeros(size(q, 1), 18);

%%%%%%%%%% get angles
for i = 1 : 18
```

```matlab
    angles(:, i) = cumtrapz(q(:, 1), q(:, i + 19));
end

%% Plot the angles
for m = 1 : 5

    % Plot the gravity tilt angles according to accel
    figure;
    subplot(2, 1, 1), plot(q(:, 1), psi(:, 1), 'r', 'LineWidth', 1,...
        'DisplayName', ['Tilt Angles of the ' IMUNames{1} ' IMU']);
    xbounds = xlim();
    set(gca, 'xtick', xbounds(1):0.2:xbounds(2));
    a = axis;
    hold on
    plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 1,...
        'DisplayName', 'Falling Starting Time');
    plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1,...
        'DisplayName', 'Impact Time');
    plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1,...
        'DisplayName', 'Impact Ending Time');
    hold off
    ylabel('Degree');
    legend('show')
    title(['Tilt-' IMUNames{1} '(Angle between z-axis of ' IMUNames{1}...
        ' IMU and universal Z-axis)']);
    subplot(2, 1, 2), plot(q(:, 1), psi(:, m + 1), 'b', 'LineWidth', 1,...
        'DisplayName', ['Tilt Angles of the ' IMUNames{m + 1} ' IMU']);
    xbounds = xlim();
    set(gca, 'xtick', xbounds(1):0.2:xbounds(2));
    a = axis;
    hold on
    plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 1,...
        'DisplayName', 'Falling Starting Time');
```

```
    plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1,...
        'DisplayName', 'Impact Time');
    plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1,...
        'DisplayName', 'Impact Ending Time');
    hold off
    xlabel('Time (s)');
    ylabel('Degree');
    legend('show')
    title(['Tilt-' IMUNames{m + 1} '(Angle between z-axis of ' IMUNames{m + 1}...
        ' IMU and universal Z-axis)']);

    % Plot the gravity tilt angles according to gyro

    figure;
    subplot(2, 1, 1), plot(q(:, 1), angles(:, 1), 'r', 'LineWidth', 1,...
        'DisplayName', ['Tilt Angles of the ' IMUNames{1} ' IMU']);
    xbounds = xlim();
    set(gca, 'xtick', xbounds(1):0.2:xbounds(2));
    a = axis;
    hold on
    plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 1,...
        'DisplayName', 'Falling Starting Time');
    plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1,...
        'DisplayName', 'Impact Time');
    plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1,...
        'DisplayName', 'Impact Ending Time');
    hold off
    ylabel('Degree');
    legend('show')
    title(['Tilt-' IMUNames{1} ' (Angle between z-axis of ' IMUNames{1}...
        ' IMU and universal Z-axis)']);
```

```
    subplot(2, 1, 2), plot(q(:, 1), angles(:, 3*m + 1), 'b', 'LineWidth', 1,...
        'DisplayName', ['Tilt Angles of the ' IMUNames{m + 1} ' IMU']);
    xbounds = xlim();
    set(gca, 'xtick', xbounds(1):0.2:xbounds(2));
    a = axis;
    hold on
    plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 1,...
        'DisplayName', 'Falling Starting Time');
    plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1,...
        'DisplayName', 'Impact Time');
    plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1,...
        'DisplayName', 'Impact Ending Time');
    hold off
    xlabel('Time (s)');
    ylabel('Degree');
    legend('show')
    title(['Tilt-' IMUNames{m + 1} ' (Angle between z-axis of ' IMUNames{m + 1}...
        ' IMU and universal Z-axis)']);

end


%% Implement a complementary filter to calculate tilt angle

tau = impactEndSec - fallingStartSec; % time constant tau

alpha = tau / (tau + dt) * 0.99; % filter constant

tilt_compl = zeros(size(angles, 1), 6);

% Initialize the starting angles according to the angles calculated from
```

```
% the accelerometer data. Since gravity is projected to each of the 3 axes
% of the accel independently when stationary, the initial tilt angles can
% be calculated from the accel data.
tilt_skull_initial = mean(psi_skull(1 : fallingStartIdx - 300, 1)); % x_gyro
    considered to be horizontal
tilt_rightEye_initial = mean(psi_rightEye(1 : fallingStartIdx - 300, 1));
tilt_leftEye_initial = mean(psi_leftEye(1 : fallingStartIdx - 300, 1));
tilt_frontBrain_initial = mean(psi_frontBrain(1 : fallingStartIdx - 300, 1));
tilt_middleBrain_initial = mean(psi_middleBrain(1 : fallingStartIdx - 300, 1))
    ;
tilt_posteriorBrain_initial = mean(psi_posteriorBrain(1 : fallingStartIdx -
    300, 1));

tilt_initial = [tilt_skull_initial, tilt_rightEye_initial,
    tilt_leftEye_initial,...
     tilt_frontBrain_initial, tilt_middleBrain_initial,
    tilt_posteriorBrain_initial];

%%%%%%%%%% tilt angle = alpha * gyro x angles + accel y * (1 - alpha)

for i = 1 : 6
    % for the eye sensor
    for j = 1 : size(angles, 1)
        tilt_compl(j, i) = (angles(j, 3*i - 2) + tilt_initial(i)) .* alpha...
            + psi(j, i) .* (1 - alpha);

    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Note: The plot axes boundaries have to  %%%
%%% be changed for different data sections. %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% Tilt Right Eye %%%

scopeN = 1000;

figure;
subplot(2, 1, 1), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1),...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1) + tilt_initial(1), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -5, 20]) % for Skull
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 1), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), 'k--', 'LineWidth', ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{1} ' (Angle between z-axis of ' IMUNames{1}...
    ' IMU and universal Z-axis)']);
```

```

subplot(2, 1, 2), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 4) + tilt_initial(2), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, 25, 50]) % for Right Eye
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 2),...
    'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 2), 'k--', 'LineWidth', 2, 'DisplayName', ...
    'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{2} ' (Angle between z-axis of ' IMUNames{2}...
    ' IMU and universal Z-axis)']);

%%% Tilt Left Eye %%%

```

```
figure;
subplot(2, 1, 1), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1) + tilt_initial(1), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -5, 20]) % for Skull
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 1), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), 'k--', 'LineWidth', ...
    2, 'DisplayName','Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{1} ' (Angle between z-axis of ' IMUNames{1}...
    ' IMU and universal Z-axis)']);

subplot(2, 1, 2), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 7) + tilt_initial(3), ...
```

```
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, 25, 50]) % for Left Eye
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(
    fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 3), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 3), 'k--', 'LineWidth',
    ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{3} ' (Angle between z-axis of ' IMUNames{3}...
    ' IMU and universal Z-axis)']);

%%% Tilt Front Brain %%%

figure;
subplot(2, 1, 1), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1),
    ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1) + tilt_initial
    (1), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -5, 20]) % for Skull
```

```
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(
    fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 1), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), 'k--', 'LineWidth',
    ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{1} ' (Angle between z-axis of ' IMUNames{1}...
    ' IMU and universal Z-axis)']);

subplot(2, 1, 2), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1),
     ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 10) +
    tilt_initial(4), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, 5, 25]) % for Front Brain
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(
    fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 4), 'b', 'DisplayName', 'Accel Angles');
```

```
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 4), 'k--', 'LineWidth', ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{4} ' (Angle between z-axis of ' IMUNames{4}...
    ' IMU and universal Z-axis)']);

%%% Tilt Middle Brain %%%

figure;
subplot(2, 1, 1), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1) + tilt_initial(1), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -5, 20]) % for section Skull
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 1), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
```

```
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), 'k--', 'LineWidth',
    ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{1} ' (Angle between z-axis of ' IMUNames{1}...
    ' IMU and universal Z-axis)']);

subplot(2, 1, 2), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1),
    ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 13) +
    tilt_initial(5), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -15, 15]) % for Middle Brain
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(
    fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 5), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 5), 'k--', 'LineWidth',
    ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
```

```matlab
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{5} ' (Angle between z-axis of ' IMUNames{5}...
    ' IMU and universal Z-axis)']);

%%% Tilt Posterior Brain %%%

figure;
subplot(2, 1, 1), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1) + tilt_initial(1), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -5, 20]) % for Skull
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 1), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), 'k--', 'LineWidth', ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
```

```
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{1} ' (Angle between z-axis of ' IMUNames{1}...
    ' IMU and universal Z-axis)']);

subplot(2, 1, 2), plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), ...
    angles(fallingStartIdx - scopeN : impactEndIdx + scopeN, 16) + tilt_initial(6), ...
    'r', 'DisplayName', 'Gyro Angles');
axis([-inf, inf, -10, 20]) % for Posterior Brain
a = axis;
hold on
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), psi(fallingStartIdx ...
    - scopeN : impactEndIdx + scopeN, 6), 'b', 'DisplayName', 'Accel Angles');
plot(q(fallingStartIdx - scopeN : impactEndIdx + scopeN, 1), tilt_compl(...
    fallingStartIdx - scopeN : impactEndIdx + scopeN, 6), 'k--', 'LineWidth', ...
    2, 'DisplayName', 'Complementary Filtered Angles');
plot([tBefore(end), tBefore(end)],[a(3), a(4)], 'r-.', 'LineWidth', 2,...
    'DisplayName', 'Falling Starting Time');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 2,...
    'DisplayName', 'Impact Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 2,...
    'DisplayName', 'Impact Ending Time');
hold off
xlabel('Time (s)');
ylabel('Degree');
```

```
legend('show', 'Location', 'southwest')
title(['Tilt-' IMUNames{6} ' (Angle between z-axis of ' IMUNames{6}...
    ' IMU and universal Z-axis)']);


%% Calculate, filter, and plot the velocities and displacements

tailCutter = round(0.05 / dt);
tail = after(1 : tailCutter, :);
tTail = tAfter(1 : tailCutter, :);
tail_add_on = tailCutter + 120;
tail_long = after(1 : tail_add_on, :);
tTail_long = tAfter(1 : tail_add_on, :);
data = [falling; impact; tail];
t_data = [tFalling; tImpact; tTail];
t_data_long = [tFalling; tImpact; tTail_long];

% Get velocity

[v, headCut] = min(abs(tFalling - fallingStartSec));
[v, tailCut_0] = min(abs(tImpact - 0.4));
tailCut = tailCut_0 + size(tFalling, 1);
vel = [];
figure;
for i = 1 : 18
    vel = [vel, cumtrapz(t_data(headCut : tailCut), q(headCut : tailCut, i + 1))];
    subplot(6, 3, i), plot(t_data(headCut : tailCut), vel(:, i), 'b');
    a = axis;
    hold on
    plot([impactZeroSec,impactZeroSec], [a(3), a(4)], 'g');
    hold off
    xlabel('Time (s)');
```

```
    ylabel('Velocity (m/s)');
    title(dataNames{i});
end
set(gcf, 'name', [DATA_SD1 ': Velocity'], 'numbertitle', 'off')

% Get displacement
displace=[];
figure;
for i = 1 : 18
    displace = [displace, cumtrapz(t_data(headCut : tailCut), vel(:, i))];
    subplot(6, 3, i),plot(t_data(headCut : tailCut), displace(:, i), 'r');
    a = axis;
    hold on
    plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g');
    hold off
    xlabel('Time (s)');
    ylabel('Displacement (m)');
    title(dataNames{i});
end
set(gcf,'name',[DATA_SD1 ': Displacement'], 'numbertitle', 'off')

%% Filter and plot the data of the universal coordinate system

Z_skull = Z_actual_skull ./ cosd(tilt_compl(:, 1));
Z_rightEye = Z_actual_rightEye ./ cosd(tilt_compl(:, 2));
Z_leftEye = Z_actual_leftEye ./ cosd(tilt_compl(:, 3));
Z_frontBrain = Z_actual_frontBrain ./ cosd(tilt_compl(:, 4));
Z_middleBrain = Z_actual_middleBrain ./ cosd(tilt_compl(:, 5));
Z_posteriorBrain = Z_actual_posteriorBrain ./ cosd(tilt_compl(:, 6));

Z_IMU = [Z_skull, Z_rightEye, Z_leftEye, Z_frontBrain, Z_middleBrain, ...
    Z_posteriorBrain];

```

```
%% Overall accelerations, velocities and displacements along the universal Z-axis

vel2_skull = [];
displace2_skull = [];
vel2_rightEye = [];
displace2_rightEye = [];
vel2_leftEye = [];
displace2_leftEye = [];
vel2_frontBrain = [];
displace2_frontBrain = [];
vel2_middleBrain = [];
displace2_middleBrain = [];
vel2_posteriorBrain = [];
displace2_posteriorBrain = [];

vel_ur1 = [];
displace_ur1 = [];
vel_ur2 = [];
displace_ur2 = [];
vel_ur3 = [];
displace_ur3 = [];
vel_ur4 = [];
displace_ur4 = [];
vel_ur5 = [];
displace_ur5 = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Note: The plot axes boundaries have to  %%%
%%% be changed for different data sections. %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% LPF the accelerations in universal coordinat system %%%
```

```
Z_skull = filtfilt(bL, aL, Z_skull);
Z_rightEye = filtfilt(bL, aL, Z_rightEye);
Z_leftEye = filtfilt(bL, aL, Z_leftEye);

universal_rightEye_accel = Z_rightEye - Z_skull;
universal_leftEye_accel = Z_leftEye - Z_skull;
universal_frontBrain_accel = Z_frontBrain - Z_skull;
universal_middleBrain_accel = Z_middleBrain - Z_skull;
universal_posteriorBrain_accel = Z_posteriorBrain - Z_skull;

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Figure for Right Eye %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure;

subplot(3, 3, 1), plot(t_data, Z_skull(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);

hold off
ylabel('Acceleration (m/s^2)');
str1 = sprintf('Acceleration of skull in universal Z-axis');
title(str1);
grid on;
set(gcf, 'name', [DATA_SD1 ': Overall accelerations, velocities and
    displacements during impact'],'numbertitle','off')
% Velocity
```

```
vel2_skull = [vel2_skull; cumtrapz(t_data_1(fallingStartIdx+1 : impactEndIdx, :),...
    Z_skull(fallingStartIdx+1 : impactEndIdx, :)); cumtrapz(t_data_1(impactEndIdx ...
    + 1: impactEndIdx + tailCutter), Z_skull(impactEndIdx + 1 : impactEndIdx ...
    + tailCutter, :))];
subplot(3, 3, 2), plot(t_data, vel2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str2 = sprintf('Skull velocity, fall time = %f s', fall_time);
title(str2);
grid on;
% Displacement
displace2_skull = [displace2_skull, cumtrapz(t_data, vel2_skull)];
subplot(3, 3, 3),plot(t_data, displace2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str3 = sprintf('Skull displacement');
```

```
title(str3);
grid on;


subplot(3, 3, 4), plot(t_data, Z_rightEye(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Acceleration (m/s^2)');
str4 = sprintf('Acceleration of right eye in universal Z-axis');
title(str4);
grid on;
% Velocity
vel2_rightEye = [vel2_rightEye; cumtrapz(t_data_1(fallingStartIdx+1 :
    impactEndIdx, :),...
    Z_rightEye(fallingStartIdx+1 : impactEndIdx, :)); cumtrapz(t_data_1(
    impactEndIdx ...
    + 1: impactEndIdx + tailCutter), Z_rightEye(impactEndIdx + 1 :
    impactEndIdx ...
    + tailCutter, :))];
subplot(3, 3, 5),plot(t_data, vel2_rightEye, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
```

```
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str5 = sprintf('Right eye velocity');
title(str5);
grid on;
% Displacement
displace2_rightEye = [displace2_rightEye, cumtrapz(t_data, vel2_rightEye)];
subplot(3, 3, 6),plot(t_data, displace2_rightEye, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str6 = sprintf('Right eye displacement');
title(str6);
grid on;

subplot(3, 3, 7), plot(t_data, universal_rightEye_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
```

```
ylabel('Acceleration (m/s^2)');
title('Acceleration of right eye relative to skull in universal Z-axis');
grid on;
% Velocity
vel_ur1 = [vel_ur1, cumtrapz(t_data, universal_rightEye_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :))];
subplot(3, 3, 8), plot(t_data, vel2_rightEye - vel2_skull, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.9, 0.5]) %for section 1
% axis([t_data(1), t_data(end), -0.25, 0.2]) %for section 2
% axis([t_data(1), t_data(end), -0.3, 0.1]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Velocity (m/s)');
str7 = sprintf('Velocity of right eye relative to skull');
title(str7);
grid on;
% Displacement
displace_ur1 = displace2_rightEye - displace2_skull;
subplot(3, 3, 9), plot(t_data, displace_ur1, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.02, 0.01]) %for section 1
% axis([t_data(1), t_data(end), -0.015, 0.001]) %for section 2
% axis([t_data(1), t_data(end), -0.02, 0.001]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
```

```
xlabel('Time (s)');
ylabel('Displacement (m)');
str8 = sprintf('Displacement of right eye relative to skull');
title(str8);
grid on;


%%%%%%%%%%%%%%%%%%%%%%%%
%%% Figure for Left Eye %%%
%%%%%%%%%%%%%%%%%%%%%%%%

figure;

subplot(3, 3, 1), plot(t_data, Z_skull(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);

hold off
ylabel('Acceleration (m/s^2)');
str1 = sprintf('Acceleration of skull in universal Z-axis');
title(str1);
grid on;
set(gcf, 'name', [DATA_SD1 ': Overall accelerations, velocities and
    displacements during impact'],'numbertitle','off')
% Velocity
subplot(3, 3, 2), plot(t_data, vel2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
```

```
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str2 = sprintf('Skull velocity, fall time = %f s', fall_time);
title(str2);
grid on;
% Displacement
subplot(3, 3, 3),plot(t_data, displace2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str3 = sprintf('Skull displacement');
title(str3);
grid on;


subplot(3, 3, 4), plot(t_data, Z_leftEye(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
```

```
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Acceleration (m/s^2)');
str4 = sprintf('Acceleration of left eye in universal Z-axis');
title(str4);
grid on;
% Velocity
vel2_leftEye = [vel2_leftEye; cumtrapz(t_data_1(fallingStartIdx+1 :
    impactEndIdx, :),...
     Z_leftEye(fallingStartIdx+1 : impactEndIdx, :)); cumtrapz(t_data_1(
    impactEndIdx ...
    + 1: impactEndIdx + tailCutter), Z_leftEye(impactEndIdx + 1 : impactEndIdx
     ...
    + tailCutter, :))];
subplot(3, 3, 5),plot(t_data, vel2_leftEye, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str5 = sprintf('Left eye velocity');
title(str5);
grid on;
% Displacement
displace2_leftEye = [displace2_leftEye, cumtrapz(t_data, vel2_leftEye)];
subplot(3, 3, 6),plot(t_data, displace2_leftEye, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
```

```
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str6 = sprintf('Left eye displacement');
title(str6);
grid on;

subplot(3, 3, 7), plot(t_data, universal_leftEye_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration of left eye relative to skull in universal Z-axis');
grid on;
% Velocity
vel_ur2 = [vel_ur2, cumtrapz(t_data, universal_leftEye_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :))];
subplot(3, 3, 8), plot(t_data, vel2_leftEye - vel2_skull, 'r', 'LineWidth', 1);
% subplot(3, 3, 8), plot(t_data, vel_ur2, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.9, 0.5]) %for section 1
```

```matlab
% axis([t_data(1), t_data(end), -0.25, 0.2]) %for section 2
% axis([t_data(1), t_data(end), -0.3, 0.1]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Velocity (m/s)');
str7 = sprintf('Velocity of left eye relative to skull');
title(str7);
grid on;
% Displacement
displace_ur2 = displace2_leftEye - displace2_skull;
subplot(3, 3, 9), plot(t_data, displace_ur2, 'r', 'LineWidth', 1);
hold on
% plot(t_data, displace_ur, 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.02, 0.01]) %for section 1
% axis([t_data(1), t_data(end), -0.015, 0.001]) %for section 2
% axis([t_data(1), t_data(end), -0.02, 0.001]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Displacement (m)');
str8 = sprintf('Displacement of left eye relative to skull');
title(str8);
grid on;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Figure for Front Brain %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

figure;

subplot(3, 3, 1), plot(t_data, Z_skull(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);

hold off
ylabel('Acceleration (m/s^2)');
str1 = sprintf('Acceleration of skull in universal Z-axis');
title(str1);
grid on;
set(gcf, 'name', [DATA_SD1 ': Overall accelerations, velocities and displacements during impact'],'numbertitle','off')
% Velocity
subplot(3, 3, 2), plot(t_data, vel2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str2 = sprintf('Skull velocity, fall time = %f s', fall_time);
title(str2);
grid on;
% Displacement
```

```
subplot(3, 3, 3),plot(t_data, displace2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str3 = sprintf('Skull displacement');
title(str3);
grid on;


subplot(3, 3, 4), plot(t_data, Z_frontBrain(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Acceleration (m/s^2)');
str4 = sprintf('Acceleration of front brain in universal Z-axis');
title(str4);
grid on;
% Velocity
vel2_frontBrain = [vel2_frontBrain; cumtrapz(t_data_1(fallingStartIdx+1 : impactEndIdx, :) ,...
```

```
    Z_frontBrain(fallingStartIdx+1 : impactEndIdx, :)); cumtrapz(t_data_1(impactEndIdx ...
    + 1: impactEndIdx + tailCutter), Z_frontBrain(impactEndIdx + 1 : impactEndIdx ...
    + tailCutter, :))];
subplot(3, 3, 5),plot(t_data, vel2_frontBrain, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str5 = sprintf('Front brain velocity');
title(str5);
grid on;
% Displacement
displace2_frontBrain = [displace2_frontBrain, cumtrapz(t_data, vel2_frontBrain)];
subplot(3, 3, 6),plot(t_data, displace2_frontBrain, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str6 = sprintf('Front brain displacement');
title(str6);
```

```
grid on;

subplot(3, 3, 7), plot(t_data, universal_frontBrain_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration of front brain relative to skull in universal Z-axis');
grid on;
% Velocity
vel_ur3 = [vel_ur3, cumtrapz(t_data, universal_frontBrain_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :))];
subplot(3, 3, 8), plot(t_data, vel_ur3, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.9, 0.5]) %for section 1
% axis([t_data(1), t_data(end), -0.25, 0.2]) %for section 2
% axis([t_data(1), t_data(end), -0.3, 0.1]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Velocity (m/s)');
str7 = sprintf('Velocity of front brain relative to skull');
```

```
title(str7);
grid on;
% Displacement
displace_ur3 = displace2_frontBrain - displace2_skull;
subplot(3, 3, 9), plot(t_data, displace_ur3, 'r', 'LineWidth', 1);
hold on
% plot(t_data, displace_ur, 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.02, 0.01]) %for section 1
% axis([t_data(1), t_data(end), -0.015, 0.001]) %for section 2
% axis([t_data(1), t_data(end), -0.02, 0.001]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Displacement (m)');
str8 = sprintf('Displacement of front brain relative to skull');
title(str8);
grid on;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Figure for Middle Brain %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure;

subplot(3, 3, 1), plot(t_data, Z_skull(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
```

```matlab
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);

hold off
ylabel('Acceleration (m/s^2)');
str1 = sprintf('Acceleration of skull in universal Z-axis');
title(str1);
grid on;
set(gcf, 'name', [DATA_SD1 ': Overall accelerations, velocities and
    displacements during impact'],'numbertitle','off')
% Velocity
subplot(3, 3, 2), plot(t_data, vel2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str2 = sprintf('Skull velocity, fall time = %f s', fall_time);
title(str2);
grid on;
% Displacement
subplot(3, 3, 3),plot(t_data, displace2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
```

```
ylabel('Displacement (m)');
str3 = sprintf('Skull displacement');
title(str3);
grid on;


subplot(3, 3, 4), plot(t_data, Z_middleBrain(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -150, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Acceleration (m/s^2)');
str4 = sprintf('Acceleration of middle brain in universal Z-axis');
title(str4);
grid on;
% Velocity
vel2_middleBrain = [vel2_middleBrain; cumtrapz(t_data_1(fallingStartIdx+1 : impactEndIdx, :),...
    Z_middleBrain(fallingStartIdx+1 : impactEndIdx, :)); cumtrapz(t_data_1(impactEndIdx ...
    + 1: impactEndIdx + tailCutter), Z_middleBrain(impactEndIdx + 1 : impactEndIdx ...
    + tailCutter, :))];
subplot(3, 3, 5),plot(t_data, vel2_middleBrain, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
```

```
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str5 = sprintf('Middle brain velocity');
title(str5);
grid on;
% Displacement
displace2_middleBrain = [displace2_middleBrain, cumtrapz(t_data, vel2_middleBrain)];
subplot(3, 3, 6),plot(t_data, displace2_middleBrain, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str6 = sprintf('Middle brain displacement');
title(str6);
grid on;

subplot(3, 3, 7), plot(t_data, universal_middleBrain_accel(fallingStartIdx+1:...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -150, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
```

```
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration of middle brain relative to skull in universal Z-axis');
grid on;
% Velocity
vel_ur4 = [vel_ur4, cumtrapz(t_data, universal_middleBrain_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :))];
subplot(3, 3, 8), plot(t_data, vel_ur4, 'r', 'LineWidth', 1);
hold on
% plot(t_data, vel_ur, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.9, 0.5]) %for section 1
% axis([t_data(1), t_data(end), -0.25, 0.2]) %for section 2
% axis([t_data(1), t_data(end), -0.3, 0.1]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Velocity (m/s)');
str7 = sprintf('Velocity of middle brain relative to skull');
title(str7);
grid on;
% Displacement
displace_ur4 = displace2_middleBrain - displace2_skull;
subplot(3, 3, 9), plot(t_data, displace_ur4, 'r', 'LineWidth', 1);
hold on
% plot(t_data, displace_ur, 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.02, 0.01]) %for section 1
% axis([t_data(1), t_data(end), -0.015, 0.001]) %for section 2
```

```
% axis([t_data(1), t_data(end), -0.02, 0.001]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Displacement (m)');
str8 = sprintf('Displacement of middle brain relative to skull');
title(str8);
grid on;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Figure for Posterior Brain %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure;

subplot(3, 3, 1), plot(t_data, Z_skull(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for section 1
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);

hold off
ylabel('Acceleration (m/s^2)');
str1 = sprintf('Acceleration of skull in universal Z-axis');
title(str1);
grid on;
set(gcf, 'name', [DATA_SD1 ': Overall accelerations, velocities and
    displacements during impact'],'numbertitle','off')
```

```
% Velocity
subplot(3, 3, 2), plot(t_data, vel2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str2 = sprintf('Skull velocity, fall time = %f s', fall_time);
title(str2);
grid on;
% Displacement
subplot(3, 3, 3),plot(t_data, displace2_skull, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str3 = sprintf('Skull displacement');
title(str3);
grid on;


subplot(3, 3, 4), plot(t_data, Z_posteriorBrain(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -150, 150]) %for section 1
```

```
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Acceleration (m/s^2)');
str4 = sprintf('Acceleration of posterior brain in universal Z-axis');
title(str4);
grid on;
% Velocity
vel2_posteriorBrain = [vel2_posteriorBrain; cumtrapz(t_data_1(fallingStartIdx+1 : impactEndIdx, :) ,...
    Z_posteriorBrain(fallingStartIdx+1 : impactEndIdx, :)); cumtrapz(t_data_1(impactEndIdx ...
    + 1: impactEndIdx + tailCutter), Z_posteriorBrain(impactEndIdx + 1 : impactEndIdx ...
    + tailCutter, :))];
subplot(3, 3, 5),plot(t_data, vel2_posteriorBrain, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -1, 1.5]) %for section 1
% axis([t_data(1), t_data(end), -1, 1]) %for section 2
% axis([t_data(1), t_data(end), -1, 1]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Velocity (m/s)');
str5 = sprintf('Posterior brain velocity');
title(str5);
grid on;
% Displacement
```

```
displace2_posteriorBrain = [displace2_posteriorBrain, cumtrapz(t_data, vel2_posteriorBrain)];
subplot(3, 3, 6),plot(t_data, displace2_posteriorBrain, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.04, 0.02]) %for section 1
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 2
% axis([t_data(1), t_data(end), -0.045, 0.005]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Displacement (m)');
str6 = sprintf('Posterior brain displacement');
title(str6);
grid on;

subplot(3, 3, 7), plot(t_data, universal_posteriorBrain_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -150, 150]) %for section 1
% axis([t_data(1), t_data(end), -50, 200]) %for section 2
% axis([t_data(1), t_data(end), -100, 200]) %for section 3
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration of posterior brain relative to skull in universal Z-axis');
grid on;
% Velocity
```

```
vel_ur5 = [vel_ur5, cumtrapz(t_data, universal_posteriorBrain_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :))];
subplot(3, 3, 8), plot(t_data, vel_ur5, 'r', 'LineWidth', 1);
hold on
% plot(t_data, vel_ur, 'r', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.9, 0.5]) %for section 1
% axis([t_data(1), t_data(end), -0.25, 0.2]) %for section 2
% axis([t_data(1), t_data(end), -0.3, 0.1]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Velocity (m/s)');
str7 = sprintf('Velocity of posterior brain relative to skull');
title(str7);
grid on;
% Displacement
displace_ur5 = displace2_posteriorBrain - displace2_skull;
subplot(3, 3, 9), plot(t_data, displace_ur5, 'r', 'LineWidth', 1);
hold on
% plot(t_data, displace_ur, 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -0.02, 0.01]) %for section 1
% axis([t_data(1), t_data(end), -0.015, 0.001]) %for section 2
% axis([t_data(1), t_data(end), -0.02, 0.001]) %for section 3
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Displacement (m)');
str8 = sprintf('Displacement of posterior brain relative to skull');
```

```
title(str8);
grid on;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Figure for Comparisons %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure;

subplot(2, 5, 1), plot(t_data, universal_rightEye_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for Right Eye
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
ylabel('Acceleration (m/s^2)');
title('Relative Accelerations of Right Eye v.s. Skull');
grid on;

subplot(2, 5, 6), plot(t_data, displace_ur1, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.02, 0.01]) %for Right Eye
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
ylabel('Displacement (m)');
title('Relative Displacements of Right Eye v.s. Skull');
grid on;
```

```matlab

subplot(2, 5, 2), plot(t_data, universal_leftEye_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for Left Eye
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
title('Left Eye v.s. Skull');
grid on;

subplot(2, 5, 7), plot(t_data, displace_ur2, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.02, 0.01]) %for Left Eye
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
str8 = sprintf('Left Eye v.s. Skull');
title(str8);
grid on;

subplot(2, 5, 3), plot(t_data, universal_frontBrain_accel(fallingStartIdx+1 :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -100, 150]) %for Front Brain
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
```

```
title('Front Brain v.s. Skull');
grid on;

subplot(2, 5, 8), plot(t_data, displace_ur3, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.02, 0.01]) %for Front Brain
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
title('Front Brain v.s. Skull');
grid on;

subplot(2, 5, 4), plot(t_data, universal_middleBrain_accel(fallingStartIdx+1
    :...
    impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -150, 150]) %for Middle Brain
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
title('Middle Brain v.s. Skull');
grid on;

subplot(2, 5, 9), plot(t_data, displace_ur4, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.02, 0.01]) %for Middle Brain
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
```

```
xlabel('Time (s)');
title('Middle Brain v.s. Skull');
grid on;

subplot(2, 5, 5), plot(t_data, universal_posteriorBrain_accel(fallingStartIdx
    +1 :...
     impactEndIdx + tailCutter, :), 'b', 'LineWidth', 1);
axis([t_data(1), t_data(end), -150, 150]) %for Posterior Brain
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
title('Posterior Brain v.s. Skull');
grid on;

subplot(2, 5, 10), plot(t_data, displace_ur5, 'r', 'LineWidth', 1);
hold on
axis([t_data(1), t_data(end), -0.02, 0.01]) %for Posterior Brain
a = axis;
plot([impactZeroSec, impactZeroSec], [a(3), a(4)], 'g', 'LineWidth', 1.2);
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2);
hold off
xlabel('Time (s)');
title('Posterior Brain v.s. Skull');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Comparisons in One Figure %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t_20ms = round(0.02 / dt);

```

```matlab
figure;

plot(t_data, universal_rightEye_accel(fallingStartIdx+1 : impactEndIdx +...
    tailCutter, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
    'Right Eye v.s. Skull');
axis([t_data(1), t_data(end), -150, 150])
a = axis;
hold on
plot(t_data, universal_leftEye_accel(fallingStartIdx+1 : impactEndIdx +...
    tailCutter, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Left Eye v.s. Skull');
plot(t_data, universal_frontBrain_accel(fallingStartIdx+1 : impactEndIdx +...
    tailCutter, :), 'r-.', 'LineWidth', 1.2, 'DisplayName',...
    'Front Brain v.s. Skull');
plot(t_data, universal_middleBrain_accel(fallingStartIdx+1 : impactEndIdx +...
    tailCutter, :), 'g-.', 'LineWidth', 1.2, 'DisplayName',...
    'Middle Brain v.s. Skull');
plot(t_data, universal_posteriorBrain_accel(fallingStartIdx+1 : impactEndIdx +...
    tailCutter, :), 'b-.', 'LineWidth', 1.2, 'DisplayName',...
    'Posterior Brain v.s. Skull');
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--', 'LineWidth', 1.2,...
    'DisplayName', '20 ms Passive Movements End Time');
hold off
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
legend('show', 'Location', 'northwest')
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% Cross-correlations %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% RF %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_rightEye_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
    'Right Eye v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Right Eye v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_frontBrain_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Front Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
```

```
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Front Brain v.s. Skull');
[acor_rf, lag_rf] = xcorr(universal_rightEye_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_frontBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_rf] = max(abs(acor_rf));
lagDiff_rf = lag_rf(I_rf)+1;
timeDiff_rf = lagDiff_rf/Fs;
grid on
subplot(3, 1, 3), plot(lag_rf, acor_rf, 'LineWidth', 1.2);
a_rf = gca;
a_rf.XTick = sort([-500:100:500 lagDiff_rf]);
xlabel('Lag in Number of Samples');
str_rf = sprintf('Cross-correlation of Right Eye and Front Brain with Lag of %
    f s', timeDiff_rf);
title(str_rf);
grid on

%%% RM %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_rightEye_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
```

```
    'Right Eye v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Right Eye v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_middleBrain_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Middle Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
```

```
ylabel('Acceleration (m/s^2)');
title('Middle Brain v.s. Skull');
[acor_rm, lag_rm] = xcorr(universal_rightEye_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_middleBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_rm] = max(abs(acor_rm));
lagDiff_rm = lag_rm(I_rm)+1;
timeDiff_rm = lagDiff_rm/Fs;
grid on
subplot(3, 1, 3), plot(lag_rm, acor_rm, 'LineWidth', 1.2);
a_rm = gca;
a_rm.XTick = sort([-500:100:500 lagDiff_rm]);
xlabel('Lag in Number of Samples');
str_rm = sprintf('Cross-correlation of Right Eye and Middle Brain with Lag of %f s', timeDiff_rm);
title(str_rm);
grid on

%%% RP %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_rightEye_accel(fallingStartIdx+1 : impactEndIdx +...
    tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
    'Right Eye v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
```

```
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Right Eye v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_posteriorBrain_accel(fallingStartIdx+1 :
    impactEndIdx +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Posterior Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Posterior Brain v.s. Skull');
[acor_rp, lag_rp] = xcorr(universal_rightEye_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_posteriorBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_rp] = max(abs(acor_rp));
lagDiff_rp = lag_rp(I_rp);
timeDiff_rp = lagDiff_rp/Fs;
grid on
```

```
subplot(3, 1, 3), plot(lag_rp, acor_rp, 'LineWidth', 1.2);
a_rp = gca;
a_rp.XTick = sort([-500:100:500 lagDiff_rp]);
xlabel('Lag in Number of Samples');
str_rp = sprintf('Cross-correlation of Right Eye and Posterior Brain with Lag
    of %f s', timeDiff_rp);
title(str_rp);
grid on

%%% LF %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_leftEye_accel(fallingStartIdx+1 : impactEndIdx
    +...
     tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
     'Left Eye v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
     'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
     'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
     'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Left Eye v.s. Skull');
grid on
subplot(3, 1, 2),
```

```
plot(t_data_long, universal_frontBrain_accel(fallingStartIdx+1 : impactEndIdx +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Front Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Front Brain v.s. Skull');
[acor_lf, lag_lf] = xcorr(universal_leftEye_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_frontBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_lf] = max(abs(acor_lf));
lagDiff_lf = lag_lf(I_lf);
timeDiff_lf = lagDiff_lf/Fs;
grid on
subplot(3, 1, 3), plot(lag_lf, acor_lf, 'LineWidth', 1.2);
a_lf = gca;
a_lf.XTick = sort([-500:100:500 lagDiff_lf]);
xlabel('Lag in Number of Samples');
str_lf = sprintf('Cross-correlation of Left Eye and Front Brain with Lag of %f s', timeDiff_lf);
title(str_lf);
grid on
```

```
%%% LM %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_leftEye_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
    'Left Eye v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Left Eye v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_middleBrain_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Middle Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
```

```
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Middle Brain v.s. Skull');
[acor_lm, lag_lm] = xcorr(universal_leftEye_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_middleBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_lm] = max(abs(acor_lm));
lagDiff_lm = lag_lm(I_lm);
timeDiff_lm = lagDiff_lm/Fs;
grid on
subplot(3, 1, 3), plot(lag_lm, acor_lm, 'LineWidth', 1.2);
a_lm = gca;
a_lm.XTick = sort([-500:100:500 lagDiff_lm]);
xlabel('Lag in Number of Samples');
str_lm = sprintf('Cross-correlation of Left Eye and Middle Brain with Lag of %
    f s', timeDiff_lm);
title(str_lm);
grid on

%%% LP %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_leftEye_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
    'Left Eye v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
```

```
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Left Eye v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_posteriorBrain_accel(fallingStartIdx+1 : impactEndIdx +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Posterior Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Posterior Brain v.s. Skull');
```

```
[acor_lp, lag_lp] = xcorr(universal_leftEye_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_posteriorBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_lp] = max(abs(acor_lp));
lagDiff_lp = lag_lp(I_lp);
timeDiff_lp = lagDiff_lp/Fs;
grid on
subplot(3, 1, 3), plot(lag_lp, acor_lp, 'LineWidth', 1.2);
a_lp = gca;
a_lp.XTick = sort([-500:100:500 lagDiff_lp]);
xlabel('Lag in Number of Samples');
str_lp = sprintf('Cross-correlation of Left Eye and Posterior Brain with Lag
    of %f s', timeDiff_lp);
title(str_lp);
grid on

%%% FM %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_frontBrain_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
    'Front Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
```

```
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Front Brain v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_middleBrain_accel(fallingStartIdx+1 : impactEndIdx
    +...
    tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
    'Middle Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Middle Brain v.s. Skull');
[acor_fm, lag_fm] = xcorr(universal_frontBrain_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_middleBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_fm] = max(abs(acor_fm));
lagDiff_fm = lag_fm(I_fm)+1;
timeDiff_fm = lagDiff_fm/Fs;
grid on
subplot(3, 1, 3), plot(lag_fm, acor_fm, 'LineWidth', 1.2);
a_fm = gca;
```

```
a_fm.XTick = sort([-500:100:500 lagDiff_fm]);
xlabel('Lag in Number of Samples');
str_fm = sprintf('Cross-correlation of Front Brain and Middle Brain with Lag
    of %f s', timeDiff_fm);
title(str_fm);
grid on

%%% MP %%%
figure;
subplot(3, 1, 1),
plot(t_data_long, universal_middleBrain_accel(fallingStartIdx+1 : impactEndIdx
     +...
     tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
     'Middle Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
     'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
     'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
     'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Middle Brain v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_posteriorBrain_accel(fallingStartIdx+1 :
    impactEndIdx +...
     tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
```

```
    'Posterior Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
    'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
    'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Posterior Brain v.s. Skull');
[acor_mp, lag_mp] = xcorr(universal_middleBrain_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_posteriorBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_mp] = max(abs(acor_mp));
lagDiff_mp = lag_mp(I_mp);
timeDiff_mp = lagDiff_mp/Fs;
grid on
subplot(3, 1, 3), plot(lag_mp, acor_mp, 'LineWidth', 1.2);
a_mp = gca;
a_mp.XTick = sort([-500:100:500 lagDiff_mp]);
xlabel('Lag in Number of Samples');
str_mp = sprintf('Cross-correlation of Front Brain and Middle Brain with Lag of %f s', timeDiff_mp);
title(str_mp);
grid on

%%% FP %%%
figure;
```

```
subplot(3, 1, 1),
plot(t_data_long, universal_frontBrain_accel(fallingStartIdx+1 : impactEndIdx
    +...
     tail_add_on, :), 'r', 'LineWidth', 1.2, 'DisplayName',...
     'Front Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
     'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
     'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
     'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Front Brain v.s. Skull');
grid on
subplot(3, 1, 2),
plot(t_data_long, universal_posteriorBrain_accel(fallingStartIdx+1 :
    impactEndIdx +...
     tail_add_on, :), 'b', 'LineWidth', 1.2, 'DisplayName',...
     'Posterior Brain v.s. Skull');
axis([t_data_long(1), t_data_long(end), -150, 150])
a = axis;
hold on
plot([impactZeroSec, impactZeroSec],[a(3), a(4)], 'g', 'LineWidth', 1.2,...
     'DisplayName', 'Impact Time');
plot([impactZeroSec + 0.02, impactZeroSec + 0.02],[a(3), a(4)], 'b--',...
     'LineWidth', 1.2, 'DisplayName', '20 ms Passive Movements End Time');
plot([tImpact(end), tImpact(end)],[a(3), a(4)], 'r--', 'LineWidth', 1.2,...
```

```
    'DisplayName', 'Impact End Time');
hold off
legend('show', 'Location', 'northwest');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
title('Posterior Brain v.s. Skull');
[acor_fp, lag_fp] = xcorr(universal_frontBrain_accel(fallingStartIdx+1 : ...
    impactEndIdx + tail_add_on, :), universal_posteriorBrain_accel(...
    fallingStartIdx+1 : impactEndIdx + tail_add_on, :));
[~, I_fp] = max(abs(acor_fp));
lagDiff_fp = lag_fp(I_fp);
timeDiff_fp = lagDiff_fp/Fs;
grid on
subplot(3, 1, 3), plot(lag_fp, acor_fp, 'LineWidth', 1.2);
a_fp = gca;
a_fp.XTick = sort([-500:100:500 lagDiff_fp]);
xlabel('Lag in Number of Samples');
str_fp = sprintf('Cross-correlation of Front Brain and Posterior Brain with
   Lag of %f s', timeDiff_fp);
title(str_fp);
grid on

%%% store the relative accels in longer period for cross-correlation study %%%

csvwrite('03_rightEye_relativeAccl_long.csv', ...
    universal_rightEye_accel(fallingStartIdx+1 : impactEndIdx + ...
    tail_add_on, :));
csvwrite('03_leftEye_relativeAccl_long.csv', ...
    universal_leftEye_accel(fallingStartIdx+1 : impactEndIdx + ...
    tail_add_on, :));
csvwrite('03_frontBrain_relativeAccl_long.csv', ...
    universal_frontBrain_accel(fallingStartIdx+1 : impactEndIdx + ...
    tail_add_on, :));
```

```
csvwrite('03_middleBrain_relativeAccl_long.csv', ...
    universal_middleBrain_accel(fallingStartIdx+1 : impactEndIdx + ...
    tail_add_on, :));
csvwrite('03_posteriorBrain_relativeAccl_long.csv', ...
    universal_posteriorBrain_accel(fallingStartIdx+1 : impactEndIdx + ...
    tail_add_on, :));
```

B.6 Data Processing MATLAB Code for Correlation Studies.

```
clear, clc, close all

data = [
    0.196 0.798 0.955 0.095 0.973 0.968 0.836 0.365 0.402;
    0.461 0.861 0.938 0.585 0.942 0.97  0.704 0.31  0.381;
    0.458 0.457 0.965 0.48  0.423 0.963 0.479 0.500 0.299;
    0.975 0.98  0.991 0.946 0.987 0.991 0.54  0.54  0.527;
    0.859 0.358 0.873 0.842 0.604 0.993 0.685 0.206 0.268;
    0.404 0.693 0.900 0.647 0.92  0.989 0.793 0.092 0.077;
    0.654 0.399 0.779 0.807 0.597 0.961 0.081 0.082 0.245;
    0.448 0.61  0.842 0.532 0.753 0.983 0.389 0.207 0.176;
    0.394 0.754 0.875 0.573 0.93  0.99  0.837 0.113 0.149;
    0.697 0.819 0.959 0.805 0.915 0.991 0.893 0.281 0.39;
    0.463 0.525 0.885 0.638 0.711 0.976 0.321 0.095 0.083;
    0.598 0.774 0.973 0.552 0.878 0.993 0.608 0.193 0.216;
    0.779 0.868 0.973 0.795 0.845 0.991 0.551 0.594 0.392;
    0.386 0.935 0.975 0.552 0.944 0.972 0.921 0.16  0.224;
    0.900 0.908 0.98  0.895 0.901 0.995 0.887 0.363 0.388;
    0.683 0.796 0.925 0.624 0.935 0.991 0.948 0.249 0.098;
    0.483 0.828 0.96  0.529 0.904 0.991 0.797 0.06  0.231;];
compNames = {'Right Eye and Front Brain', 'Right Eye and Middle Brain', ...
    'Right Eye and Posterior Brain', 'Left Eye and Front Brain', ...
    'Left Eye and Middle Brain', 'Left Eye and Posterior Brain', ...
    'Skull and Front Brain', 'Skull and Middle Brain',...
    'Skull and Posterior Brain'};

figure;
for i = 1 : 9
    if i == 3
        subplot(3, 3, i), h = histfit(data(:, i), 7, 'beta');
        M = mean(data(:, i));
```

```
        h(1).FaceColor = [246/255 128/255 38/255];
        h(2).Color = [73/255 110/255 156/255];
        h(1).DisplayName = 'Histogram';
        h(2).DisplayName = 'Beta Distribution Fit';
        hold on
        axis([0, 1.05, 0, 15])
        a = axis;
        plot([M, M], [a(3), a(4)], 'LineStyle', '--', 'Color',...
            [73/255 110/255 156/255], 'LineWidth', 1.5,...
            'DisplayName', 'Mean Value');
        hold off
        legend('show', 'Location', 'northwest');
        title(compNames{i});
    else if i == 6
            subplot(3, 3, i), h = histfit(data(:, i), 2, 'beta');
            M = mean(data(:, i));
            h(1).FaceColor = [246/255 128/255 38/255];
            h(2).Color = [73/255 110/255 156/255];
            h(1).DisplayName = 'Histogram';
            h(2).DisplayName = 'Beta Distribution Fit';
            hold on
            axis([0, 1.05, 0, 15])
            a = axis;
            plot([M, M], [a(3), a(4)], 'LineStyle', '--', 'Color',...
                [73/255 110/255 156/255], 'LineWidth', 1.5,...
                'DisplayName', 'Mean Value');
            hold off
            legend('show', 'Location', 'northwest');
            title(compNames{i});
        else if i == 1 || i == 4 || i ==7
                subplot(3, 3, i), h = histfit(data(:, i), 20, 'beta');
                M = mean(data(:, i));
                h(1).FaceColor = [246/255 128/255 38/255];
```

```
                h(2).Color = [73/255 110/255 156/255];
                h(1).DisplayName = 'Histogram';
                h(2).DisplayName = 'Beta Distribution Fit';
                hold on
                axis([0, 1.05, 0, 15])
                a = axis;
                plot([M, M], [a(3), a(4)], 'LineStyle', '--', 'Color',...
                    [73/255 110/255 156/255], 'LineWidth', 1.5,...
                    'DisplayName', 'Mean Value');
                hold off
                ylabel('Frequency');
                legend('show', 'Location', 'northwest');
                title(compNames{i});
            else
                subplot(3, 3, i), h = histfit(data(:, i), 18, 'beta');
                M = mean(data(:, i));
                h(1).FaceColor = [246/255 128/255 38/255];
                h(2).Color = [73/255 110/255 156/255];
                h(1).DisplayName = 'Histogram';
                h(2).DisplayName = 'Beta Distribution Fit';
                hold on
                axis([0, 1.05, 0, 15])
                a = axis;
                plot([M, M], [a(3), a(4)], 'LineStyle', '--', 'Color',...
                    [73/255 110/255 156/255], 'LineWidth', 1.5,...
                    'DisplayName', 'Mean Value');
                hold off
                legend('show', 'Location', 'northwest');
                title(compNames{i});
            end
        end
    end
end
```

Appendix C

Teensy Code

```
#include <Wire.h>
#include <SparkFunLSM6DS3.h>
#include <SPI.h>
#include <SdFatConfig.h>
#include <FreeStack.h>
#include <MinimumSerial.h>
#include <SdFat.h>
#include <BlockDriver.h>
#include <SysCall.h>
SdFatSdio sd;

#define TEMP_L        0x20 // Temperature output register LSB
#define TEMP_H        0x21 // Temperature output register MSB
#define GYRO_OUTX_L   0x22 // Pitch (X) angular rate LSB
#define GYRO_OUTX_H   0x23 // Pitch (X) angular rate MSB
#define GYRO_OUTY_L   0x24 // Roll (Y) angular rate LSB
#define GYRO_OUTY_H   0x25 // Roll (Y) angular rate MSB
#define GYRO_OUTZ_L   0x26 // Yaw (Z) angular rate LSB
#define GYRO_OUTZ_H   0x27 // Yaw (Z) angular rate MSB
#define ACC_OUTX_L    0x28 // X axis linear acceleration value LSB
#define ACC_OUTX_H    0x29 // X axis linear acceleration value MSB
#define ACC_OUTY_L    0x2A // Y axis linear acceleration value LSB
#define ACC_OUTY_H    0x2B // Y axis linear acceleration value MSB
#define ACC_OUTZ_L    0x2C // Z axis linear acceleration value LSB
#define ACC_OUTZ_H    0x2D // Z axis linear acceleration value MSB

#define IMU_1         0x6B // The address of the flexible PCB IMU
```

```
#define IMU_2          0x6A // The address of the reference breakout IMU

#define ECHO_TO_SERIAL  0 // whether to echo data to serial port or not
#define WAIT_TO_START   1 // whether to wait for serial input in setup() or not

// how many milliseconds between grabbing data and logging it.
#define LOG_INTERVAL  0 // ms between entries (reduce to take more/faster data)

// how many milliseconds before writing the logged data permanently to disk
#define SYNC_INTERVAL 100 // ms between calls to flush() - to write data to the SD card
uint32_t syncTime = 0; // time of last sync()

// the digital pins that connect to the LEDs
#define greenLEDpin 13
#define redLEDpin 23
// #define blueLEDpin 32
// the PWM pins that connects to the bzzer
// #define buzzer A19
// the digital pin that connect to trigger
//#define recordPin 22
#define triggerPin 14
//#define camera 35
#define ttlRecord 16

// the logging file
File logfile;

//int IMUVal[7]; // IMU values array without FIFO or RAM to buffer
int blocks = 1; // number of data blocks to read and store
int matrixSize = 7700; // # of rows of data to be buffered in RAM
```

```
int16_t IMUVal[7700][16]; // A matrix to store buffered IMU values in the RAM
// unsigned char data[14] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }; // Temporary array to store read data
int16_t data[14];
unsigned long start_read, finished_read, elapsed_read;
unsigned long start_print, finished_print, elapsed_print; // set timers to test the reading and printing times


LSM6DS3 eyeIMU( I2C_MODE, IMU_1 );
LSM6DS3 headIMU( I2C_MODE, IMU_2 );

void error(char *str)
{
  Serial.print("error: ");
  Serial.println(str);

  // red LED indicates error
  digitalWrite(redLEDpin, HIGH);

  while(1);
}

void setup()
{

  // Over-ride default settings if desired
  eyeIMU.settings.gyroEnabled = 1;  // Can be 0 or 1
  eyeIMU.settings.gyroRange = 2000;   // Max deg/s.  Can be: 125, 245, 500, 1000, 2000
  eyeIMU.settings.gyroSampleRate = 1660;   // Hz.  Can be: 13, 26, 52, 104, 208, 416, 833, 1660
  eyeIMU.settings.gyroBandWidth = 400;  // Hz.  Can be: 50, 100, 200, 400;
```

```
eyeIMU.settings.gyroFifoEnabled = 1;  // Set to include gyro in FIFO
eyeIMU.settings.gyroFifoDecimation = 1;  // Set 1 for on /1

eyeIMU.settings.accelEnabled = 1;
eyeIMU.settings.accelRange = 16;      // Max G force readable.  Can be: 2, 4, 8, 16
eyeIMU.settings.accelSampleRate = 1660;  // Hz.  Can be: 13, 26, 52, 104, 208, 416, 833, 1660, 3330, 6660, 13330
eyeIMU.settings.accelBandWidth = 400;  // Hz.  Can be: 50, 100, 200, 400;
eyeIMU.settings.accelFifoEnabled = 1;  // Set to include accelerometer in the FIFO
eyeIMU.settings.accelFifoDecimation = 1;  // Set 1 for on /1
eyeIMU.settings.tempEnabled = 1;

//Non-basic mode settings
eyeIMU.settings.commMode = 1; // Can be modes 1, 2 or 3
// Over-ride default settings if desired
headIMU.settings.gyroEnabled = 1;  // Can be 0 or 1
headIMU.settings.gyroRange = 2000;   // Max deg/s.  Can be: 125, 245, 500, 1000, 2000
headIMU.settings.gyroSampleRate = 1660;   // Hz.  Can be: 13, 26, 52, 104, 208, 416, 833, 1660
headIMU.settings.gyroBandWidth = 400;  // Hz.  Can be: 50, 100, 200, 400;
headIMU.settings.gyroFifoEnabled = 1;  // Set to include gyro in FIFO
headIMU.settings.gyroFifoDecimation = 1;  // Set 1 for on /1

headIMU.settings.accelEnabled = 1;
headIMU.settings.accelRange = 16;      // Max G force readable.  Can be: 2, 4, 8, 16
headIMU.settings.accelSampleRate = 1660;  // Hz.  Can be: 13, 26, 52, 104, 208, 416, 833, 1660, 3330, 6660, 13330
headIMU.settings.accelBandWidth = 400;  // Hz.  Can be: 50, 100, 200, 400;
```

```
  headIMU.settings.accelFifoEnabled = 1;  // Set to include accelerometer in
    the FIFO
  headIMU.settings.accelFifoDecimation = 1;  // Set 1 for on /1
  headIMU.settings.tempEnabled = 1;

  //Non-basic mode settings
  headIMU.settings.commMode = 1; // Can be modes 1, 2 or 3

  Wire.begin();
  Serial.begin( 9600 );

  // use debugging LEDs
  pinMode( greenLEDpin, OUTPUT );
  pinMode( redLEDpin, OUTPUT );
//  pinMode( blueLEDpin, OUTPUT );

  // use buzzer
//  pinMode( buzzer, OUTPUT );
//  analogWrite( buzzer, 0 );

  // use trigger
  //pinMode( recordPin, INPUT_PULLUP );
  //pinMode( triggerPin, INPUT_PULLUP );
  digitalWrite( triggerPin, HIGH );
  pinMode( triggerPin, INPUT_PULLUP );
  //pinMode( triggerPin, INPUT );
  //analogWrite( triggerPin, 0 );
/*
  // use camera synchronizer
  pinMode( camera, OUTPUT );
  digitalWrite( camera, LOW );
*/
```

```
  // use falling/landing triggers/sensors
  pinMode( ttlRecord, INPUT );
  digitalWrite( ttlRecord, LOW );

}

void readIMU( int deviceAddress, int address, int16_t dataArray[] ) {

  int count = 0;

  Wire.beginTransmission( deviceAddress ); // Start transmission to device
  Wire.write( address ); // Register to read
  Wire.endTransmission();

  Wire.beginTransmission( deviceAddress );
  Wire.requestFrom( deviceAddress, 14 ); // Read 14 bytes continiously
   starting from the LSB of temperature
  for( count = 0; count < 14; count++ ) {
    while(!Wire.available());
    dataArray[count] = Wire.read(); // Store the 14 readings in one array
  } // Close for for()
  Wire.endTransmission();

} // CLose for readIMU()

// With RAM as buffer

void loop() {

  if( digitalRead(triggerPin) == LOW ) {

  // call .begin() to configure the IMU
    if( eyeIMU.begin() != 0 )
```

```
    {
      Serial.println( "Error starting eyeIMU." );
    }
    else
    {
      Serial.println( "\neyeIMU started." );
    }

    // call .begin() to configure the IMU
    if( headIMU.begin() != 0 )
    {
      Serial.println( "Error starting headIMU." );
    }
    else
    {
      Serial.println( "headIMU started." );
    }

    // initialize the SD card
    Serial.print( "\nInitializing SD card..." );
    // make sure that the default chip select pin is set to
    // output, even if you don't use it:
//    pinMode( chipSelect, OUTPUT );
//    digitalWrite( chipSelect, HIGH );

//  see if the card is present and can be initialized:
    if ( !sd.begin() ) {
      error( "Card failed, or not present" );
    }

    Serial.println( "card initialized." );

//    create a new file
```

```
    char filename[] = "LOGGER00.CSV";
    for ( uint8_t i = 0; i < 100; i++ ) {
      filename[6] = i/10 + '0';
      filename[7] = i%10 + '0';
      if ( ! sd.exists(filename) ) {
        // only open a new file if it doesn't exist
        logfile = sd.open( filename, FILE_WRITE );
        break;  // leave the loop
      } // Close for if
    } // Close for for

    if (! logfile) {
      error("cannot create file");
    }

    Serial.print( "Logging to: " );
    Serial.println( filename );

    Serial.println( "Processor came out of reset.\n" );
    Serial.println( "Waiting for trigger to start..." );

//    while( digitalRead(ttlRecord) == LOW ) {

      Serial.println( "Started!" );

      for (int b = 0; b < blocks; b++) {

        digitalWrite( greenLEDpin, HIGH );
        //logfile.println( RTC.now() );
        uint16_t start_read = millis();

        for ( int i = 0; i < matrixSize; i+= 2 ) {
```

```
        IMUVal[i][0] = millis(); // Mark the reading time and store in the first column of the matrix
        //IMUVal[i][8] = digitalRead(triggerPin); // Record the landing status and store in the last (9th) column of the matrix

        readIMU( IMU_1, TEMP_L, data );
        for ( int j = 0; j < 7; j++ ) {
          IMUVal[i][j+1] = ( (data[2*j+1] << 8) | data[2*j] ); // Store the read data in RAM
        } // CLose for j
        readIMU( IMU_2, TEMP_L, data );
        for ( int k = 0; k < 7; k++ ) {
          IMUVal[i][k+8] = ( (data[2*k+1] << 8) | data[2*k] ); // Store the read data in RAM
        } // CLose for k

        // Mark the impact starting and finishing time and store in the 2nd last column of the matrix
        IMUVal[i][15] = analogRead(ttlRecord);

        // generate a TTL for the camera and store in the last column of the matrix
        //digitalWrite( camera, HIGH );
/       IMUVal[i][16] = 1024;

        delayMicroseconds(110);

        IMUVal[i+1][0] = millis(); // Mark the reading time and store in the first column of the matrix
        //IMUVal[i][8] = digitalRead(triggerPin); // Record the landing status and store in the last (9th) column of the matrix

        readIMU( IMU_1, TEMP_L, data );
```

```
        for ( int j = 0; j < 7; j++ ) {
          IMUVal[i+1][j+1] = ( (data[2*j+1] << 8) | data[2*j] ); // Store the read data in RAM
        } // CLose for j
        readIMU( IMU_2, TEMP_L, data );
        for ( int k = 0; k < 7; k++ ) {
          IMUVal[i+1][k+8] = ( (data[2*k+1] << 8) | data[2*k] ); // Store the read data in RAM
        } // CLose for k

        // Mark the impact starting and finishing time and store in the 2nd last column of the matrix
        IMUVal[i+1][15] = analogRead(ttlRecord);

        // generate a TTL for the camera and store in the last column of the matrix
        //digitalWrite( camera, LOW );
//          IMUVal[i+1][16] = 0;

        delayMicroseconds(110);

      } // Close for i

      digitalWrite( greenLEDpin, LOW );

      uint16_t finished_read = millis();

      digitalWrite( redLEDpin, HIGH );
      //digitalWrite( camera, HIGH );
      //analogWrite( buzzer, 127 );

      uint16_t start_print = millis();
      for ( int p = 0; p < matrixSize; p++ ) {
```

```
          for ( int q = 0; q < 16; q++ ) {
            logfile.print( IMUVal[p][q] );
            logfile.print( "," );
          } // Close for q
        logfile.print( "\n" );
        } // Close for p

      // blink LED to show we are syncing data to the card & updating FAT!
      logfile.flush();
      uint16_t finished_print = millis();

      uint16_t elapsed_read = finished_read - start_read;
      logfile.println( elapsed_read );
      uint16_t elapsed_print = finished_print - start_print;
      logfile.println( elapsed_print );

      digitalWrite( redLEDpin, LOW );
      //digitalWrite( camera, LOW );
      //analogWrite( buzzer, 0 );

      } // Close for blocks

    logfile.close();

    Serial.println("Logging Finished.\n");

  } // Close for trigger if

} // Close for loop()
```