2020

Developing and Comparing Sensor for Movement Analysis and Biofeedback

Umaiyaal Vasudevaraja
Illinois State University, umaiyaalvasudevaraja@gmail.com

DEVELOPING AND COMPARING SENSOR FOR MOVEMENT ANALYSIS AND BIOFEEDBACK

UMAIYAAL VASUDEVARAJA

21 pages

Background: Recent advances in the field of wearable technology are now at a peak in the sports field and the medical field. The validity, reliability, and application of such systems are still under research and yet to be revealed.

Purpose: This study aims to design and constructing the Inertial Measurement Unit (IMU) hardware with the required software to collect accelerometer data for potential use in human movement studies and test the efficacy of the collected IMU accelerometer data by comparing it with the motion capture data.

Methods: In this study, the IMU sensor is coupled with the Arduino, loaded with software code used for data collection. To test the efficacy, the sensor was placed on the lumbar region during quiet standing task and an exaggerated sway of random high amplitude anteroposterior and mediolateral deviations of the model in tandem stance. A correlation analysis was conducted to assess the relationship between the measured signals as a form of comparison.

Results: The construction of the sensor was successful with certain limitations and the correlation analysis results varied for across trials. Comparisons conducted for the X and Y axes values ranged from weak to strong, while Z axis comparisons were generally weak.

Conclusions: The aims of the study were successful, although the results were not anticipated. The IMU sensor appears to be viable for biofeedback applications. However, the acceleration patterns varied across trials, which is most likely attributed to discrepancies in sampling frequency, accumulated noise, and signal processing procedures. Further research is needed to optimize data collection and processing procedures when constructing the IMU for human movement research.

Keywords: Motion Capture, Sensor, Biofeedback, Inertial Measurement Unit.

DEVELOPING AND COMPARING SENSOR FOR MOVEMENT ANALYSIS AND BIOFEEDBACK

UMAIYAAL VASUDEVARAJA

A Thesis Submitted in Partial
Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Kinesiology and Recreation

ILLINOIS STATE UNIVERSITY

2020

DEVELOPING AND COMPARING SENSOR FOR MOVEMENT ANALYSIS AND BIOFEEDBACK

UMAIYAAL VASUDEVARAJA

COMMITTEE MEMBERS:

Adam E. Jagodinsky, Chair

Rishi Saripalle

Micheal Torry

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my supervisor, Professor Dr. Adam E. Jagodinsky, who has guided and encouraged me to be professional and do the right thing even when the road got tough. Without his persistent help and support, the goal of this project would not have been realized.

I would also like to show my deepest gratitude to my professor, Dr. Rishi Saripalle, who had supported me with all the technical difficulties that I have faced in the completion of the project and guided me along the thorn path till the end, providing me with endless support and confidence.

I would also like to thank professor, Dr. Michael Torry as my class advisor and the lab assistant, Abby along with all the friends for their assistance. Special thanks to Illinois State University Graduate School for allocating the graduate research fund, for which it would have been a tough time without their help.

Nevertheless, it is lucky to meet a friend, Ares, and Mithra, in this land to help and accompany me as a family and support me overcome the difficulties with all their might. I would also like to thank Illinois state university for providing me a chance to explore the fascinating field of kinesiology and recreation.

Last but not the least important, I owe more than thanks to God, Almighty for being graceful and my family members which include my parents, my younger brother for their financial support and encouragement throughout my life. Without their support, I can't finish my college and graduate education seamlessly.

U.V.

CONTENTS

TABLES

FIGURES

CHAPTER I: INTRODUCTION

Inertial measurement units (IMU) are a widely available technology used to obtain information about the motion of an object or body. IMU technology falls under the category of Micro-Electrical-Mechanical-Systems (MEMS), which are notable for their small size, low power consumption, increased functionality, and low fabrication costs via mass productions using a modular design methodology [2]. IMUs are generally composed of an accelerometer, gyroscope, and magnetometer, used to measure the acceleration, rate of rotation (measured in degree/s), and the magnetic fields. Of these three, accelerometer markets are dominated, with silicon micromachining representing around 90% and of the makeup of accelerometer design, and the rest piezoceramic type [3].

IMU production is prevalent within many industrial sectors, yielding a wide selection of hardware components and networking platforms. The IMU sensor is widely used and is known for its accuracy, sensor support, and the low cost of the sensor measurement. When connected with the Arduino and coupled with a data logging shield consisting of an onboard SD drive and real-time capture RTC, the IMU can be efficient in collecting and storing data and timestamp. [7]. When compared with the Arduino nano new 33 BLE sense [23], the IMU sensor used is cost-effective, simple to use, and is used in a variety of applications apart from the clinical field. It is also known for its use in the research and development of biofeedback in clinical and performance fields [4]. The availability of robust and cost-effective sensor technology has opened the door for widespread use in the human movement sciences [23].

Previous applications of IMU in human movement sciences include gait analysis [8], inertial head tracker[9], post-stroke arm rehabilitation[10], arm posture correction[11], and exoskeleton design for rehabilitation[12]], etc. Hence IMU can enable us to measure the body orientation, motion, direction, and physiologic state of the moving human body, making IMU technology ideal for biofeedback applications. Biofeedback is a technique of providing biological information to a user in real-time. Biofeedback can be implemented to provide additional sensory information about body equilibrium to the brain [4]. Since IMU technology can provide a robust and cost-effective means for capturing human movement metrics, implementing IMUs for biofeedback applications in human movement studies is a plausible option.

To implement wearable IMU technology for biofeedback applications in future studies of human movement, the aim of this study is as follows: (1) to design and construct IMU hardware with the required software to collect accelerometer data for potential use during human movement studies; and (2) to test the efficacy of the custom IMU by comparing accelerometer results to motion capture results.

CHAPTER II: MATERIALS AND METHODS

A diagram outlining the integration of hardware and software elements that make up the custom IMU sensor is shown in Figure 1. The hardware is an integration of the Arduino and 9-DoF IMU sensor.

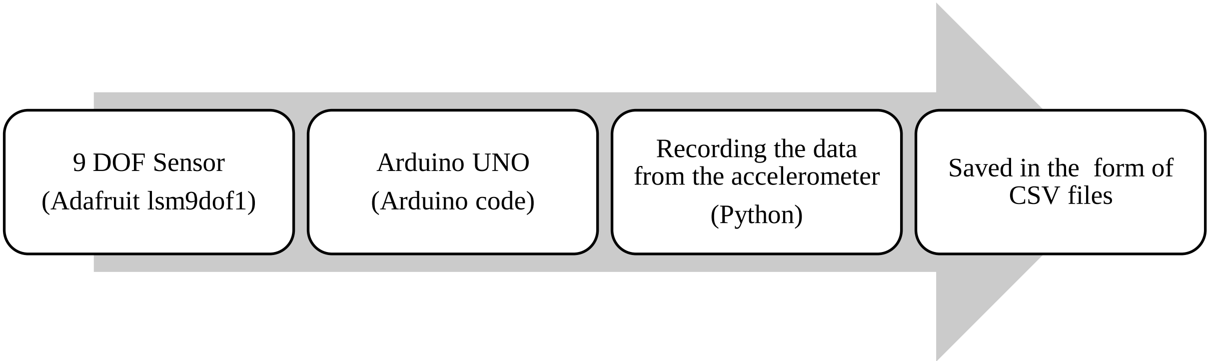

Fig 1. Flow diagram representing the sensor measurement.

Inertial Motion Sensor

The inertial Motion Sensor used here is the 9 DOF sensor, which has three sensors embedded in it. The 3 axis Accelerometer which tells us the direction down towards the earth in 3 D space. The other is a 3-axis magnetometer that can sense where the strongest magnetic force. The third is a 3 gyroscope that can measure spin and twist. This sensor combines data to project a human in 3D space.

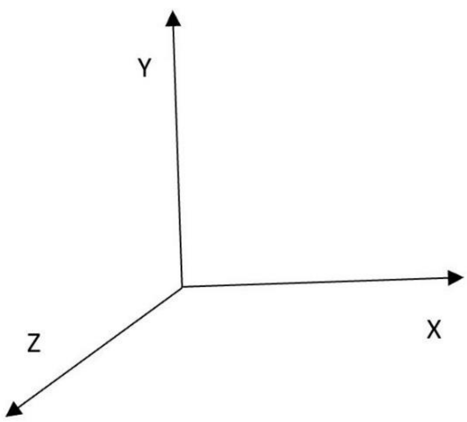

Fig 2. Inertial sensor coordinates as applied to the body.

The inertial sensor is used to calculate the acceleration of the moving human body, and Figure 2 represents the sensor coordinates as applied to the body, from the 'drift'(the accelerometer like the motion capture system first determines the position and then calculates the acceleration which are calculated inside the sensor, which causes a drift-change in position) produced in the sensor for calculating the acceleration [33].

Arduino UNO

Arduino UNO is a microcontroller board based on ATmega328P. It has 14 digital input/output pins (of which 6 can be used as Pulse width Modulation (PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller [23]. Being neither a microcontroller nor a microprocessor, Arduino UNO is simply a development board. A set of digital and analog input/output (I/O) pins is equipped in the board and it may be interfaced with various expansion boards and other circuits based on ATMEL microcontrollers which is also a microprocessor [13].

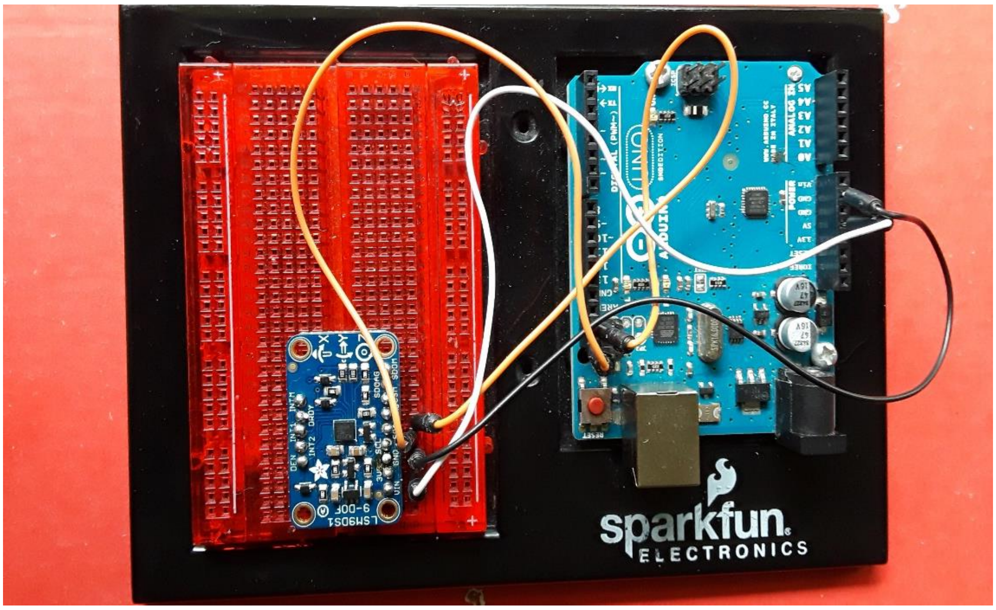

Fig 3. Connection of the Arduino with the sensor

The Arduino Uno has several facilities for communicating with a computer, another Arduino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication as shown in Figure 3, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows serial communication on any of the Uno's digital pins [23].

Software

The Arduino is programmed, using C++ (Annexure 1), based on the requirements for the data collection concerning the acceleration with the accelerometer of the LSM9DOF sensor. The sensor is placed on the body and the axis of the sensor placed is represented as in figure 2. The program is then uploaded to Arduino, the sampling frequency is set to 119 Hz, raw data of the accelerometer were calibrated, and the collected data is saved to the external storage using a python code (Annexure 2). In short, the IMU sensor is connected to the Arduino as shown in Figure 3 and the code for the Arduino program(Annexure 1) runs the LSM9DS1 sensor with a frequency of approximately 192 Hz and the measured values of the sensor are then recorded and stored in the form of datasheets with the help of the python code(Annexure 2).

Procedure

Following the construction of the IMU, the IMU was implemented to assess its accuracy. The readings of the sensor are recorded in a closed environment, i.e., the Biomechanics Lab of the School of Kinesiology and Recreation, Illinois State University.

The intended application of the custom IMU sensor is to provide feedback on a potential user of their body’s motion during common bipedal tasks. To test the viability of the custom IMU sensor for this type of application, the custom IMU sensor and gold standard optical motion-capture cameras (10 cameras, 200Hz; Vicon) were implemented during normal and exaggerated sway scenarios to allow for comparison between the two modes of measurement. A standard plug-in-gait reflective marker set was utilized to create a rigid linked-segment model representing the human body so that motion capture measurement of the model’s center of mass (COM) could be made. Furthermore, the custom IMU was attached to the lumbar region of the model to measure sway accelerations. Previous studies have indicated that the placement of the sensor on the lumbar region resulted in valid sway measurements [15-19].

Trials lasting approximately 30 seconds were conducted under two conditions: normal sway (three trials) and exaggerated sway (three trials). Normal sway trials were meant to mimic the behaviour of quiet standing. The exaggerated sway condition consisted of random high amplitude anteroposterior and mediolateral deviations of the model in tandem stance. For each trial, a motion stimulus projected along the vertical access of the model was introduced at the beginning and end of the trial to provide a point of synchronization across the motion capture and sensor datasets.

The plug-in-gait model allowed for the estimation of the model's center of mass location through built-in algorithms within the motion-capture software (Vicon Nexus). Thus, the three-dimensional position of the model's COM was collected for each trial to allow for the eventual comparison to the acceleration data collected from the custom IMU.

Post-processing of the IMU and motion capture data was conducted using a custom MATLAB code (MATLAB, MathWorks, Inc, MA, USA). First, the sensor axes are modified to match the motion capture axes such that the sensor X axis is flipped, and the Y and Z axes are interchanged. Several samples within the sensor data time series erroneously recorded as missing entries, which were removed before signal filtering. Sensor signal noise was removed using a third-order one-dimensional median filter.

The acceleration of the motion capture COM position data was calculated through double differentiation. Following differentiation, COM acceleration was filtered using a third-order median filter, and the signal was smoothed using a moving average technique with a window size of 100 frames. Following signal processing of the sensor and motion capture data, several steps were taken to facilitate comparison between the two signals. First, data of both sets are then trimmed so that both sensor and motion capture time series were aligned relative to the motion stimulus that was presented at the beginning and end of each trial. The motion capture data was then resampled to match the sampling rate of the sensor series. Finally, both sensor and motion capture data were normalized to have a common amplitude scale. Specifically, both data series were normalized to the respective maximum value. The resulting acceleration is then stored and processed (Annexure 3). Correlation analysis was conducted to indicate the spatiotemporal relationship between signals. The correlation results were interpreted as small/weak for ranges 0.1 to 0.3, medium/moderate for 0.3 to 0.5, and large/strong for 0.5 to 1.0, regardless of positive and negative linear correlation. The correlation was done three times between raw acceleration data, filtered acceleration data, and the normalized data, to explore the effect of data processing on the relationship between signals.

CHAPTER III: RESULTS

Table 1 shows the correlation of X, Y, Z-axis after normalization among the trials, and Table 2 and 3 represents the correlation of X, Y, Z-axis among all the trials pre and post data processing of Normal sway and Exaggerated sway.

Trials	Normal Sway 1	Normal Sway 2	Normal Sway 3	Exaggerated Sway 1	Exaggerated Sway 2	Exaggerated Sway 3
Correlation (in X)	-0.02	0.57	0.31	0.62	0.35	0.18
Correlation (in Y)	0.24	0.44	0.22	0.63	0.05	0.1
Correlation (in Z)	0.1	0.15	-0.13	-0.06	-0.08	-0.03

Table 1. Correlation of X, Y, Z axis in the Normal and Exaggerated Sway trials.

	Normal 1			Normal 2			Normal 3		
	RAW	Filter	Normalize	RAW	Filter	Normalize	RAW	Filter	Normalize
X	-0.01	-0.04	-0.02	0.21	0.01	0.57	0.16	0.14	0.32
Y	0	-0.03	0.24	-0.04	0.06	0.44	-0.11	-0.03	0.22
Z	0.01	-0.1	0.1	0.43	0.04	0.15	0.1	-0.13	-0.13

Table 2. Correlation of pre and post data processing for normal sway trials

	EXG 1			EXG 2			EXG 3		
	RAW	Filter	Normalize	RAW	Filter	Normalize	RAW	Filter	Normalize
X	0.52	0.61	0.62	0.31	0.33	0.35	0.14	0.18	0.18
Y	-0.36	-0.55	0.63	-0.08	-0.05	0.05	-0.03	-0.06	0.1
Z	-0.27	-0.01	-0.06	-0.2	0.02	-0.08	-0.03	-0.21	-0.03

Table 3. Correlation of pre and post data processing for exaggerated sway trials

The plotted normalized graph is depicted below from fig 4 to fig 9 for all trials of normal and exaggerated sways.

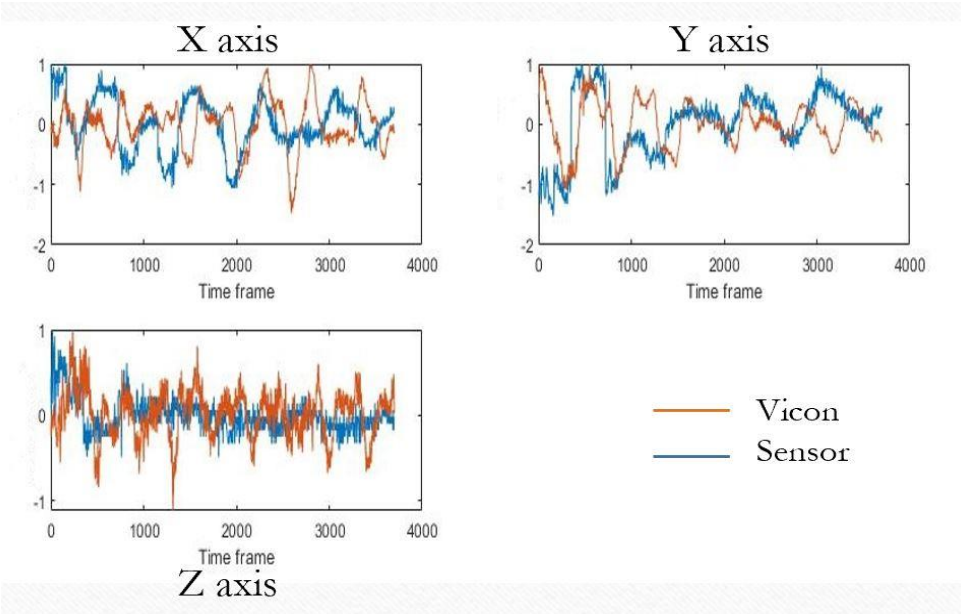

Fig 4. Normal Sway trial 1

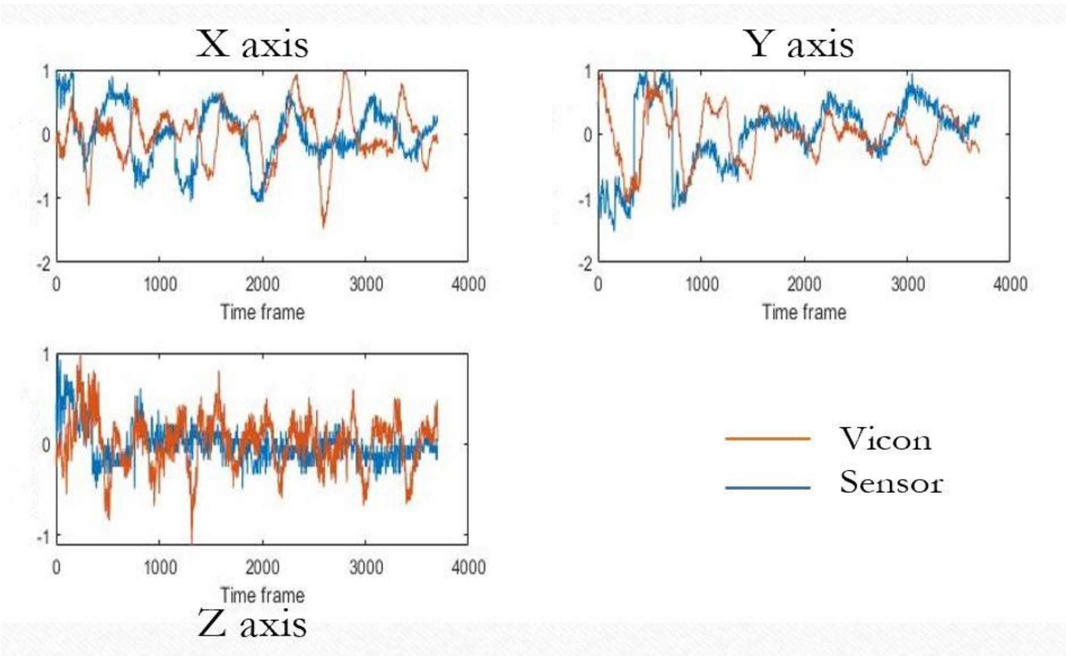

Fig 5. Normal Sway trial 2

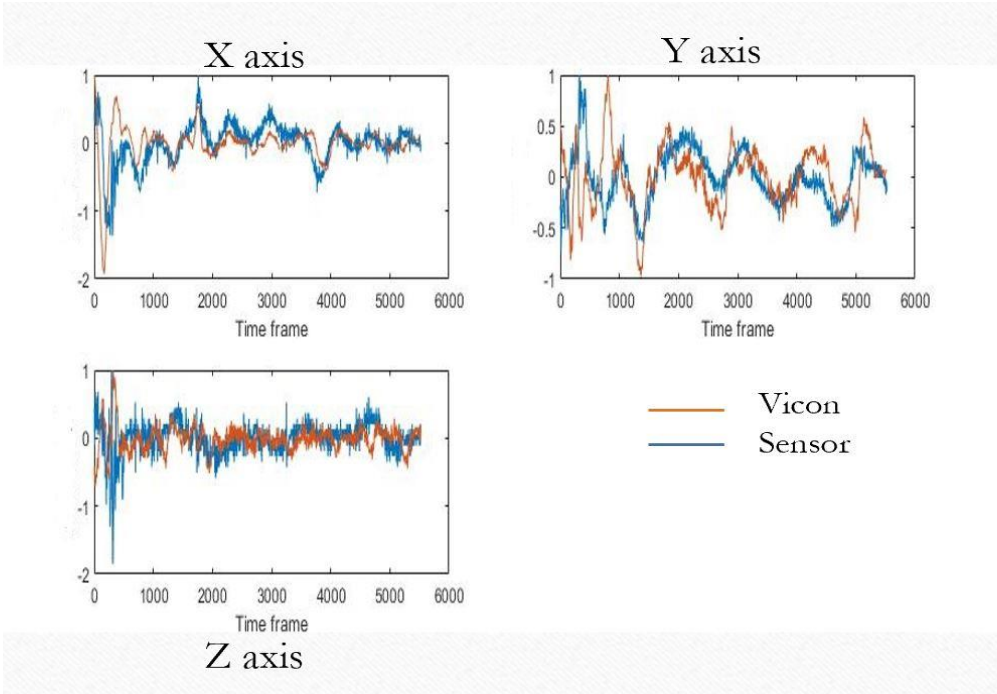

Fig 6. Normal Sway trial 3

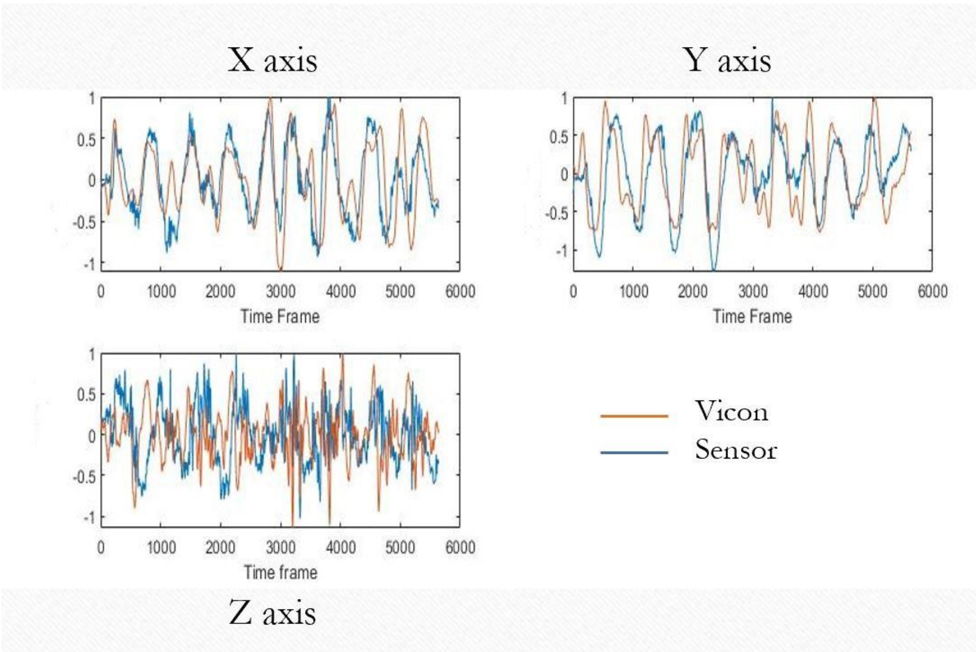

Fig 7. EXG Sway trial 1

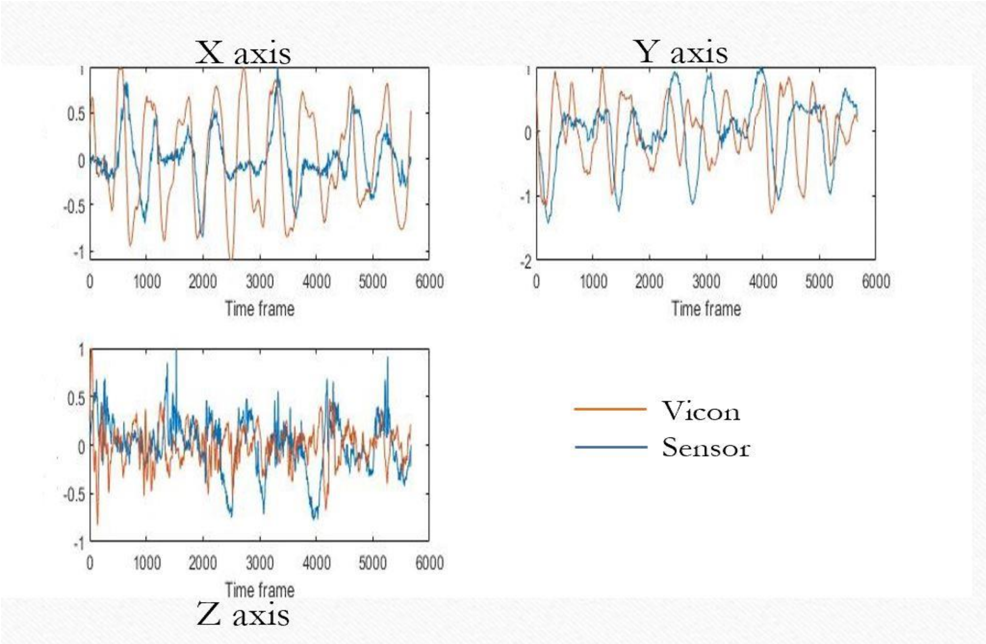

Fig 8. EXG Sway trial 2

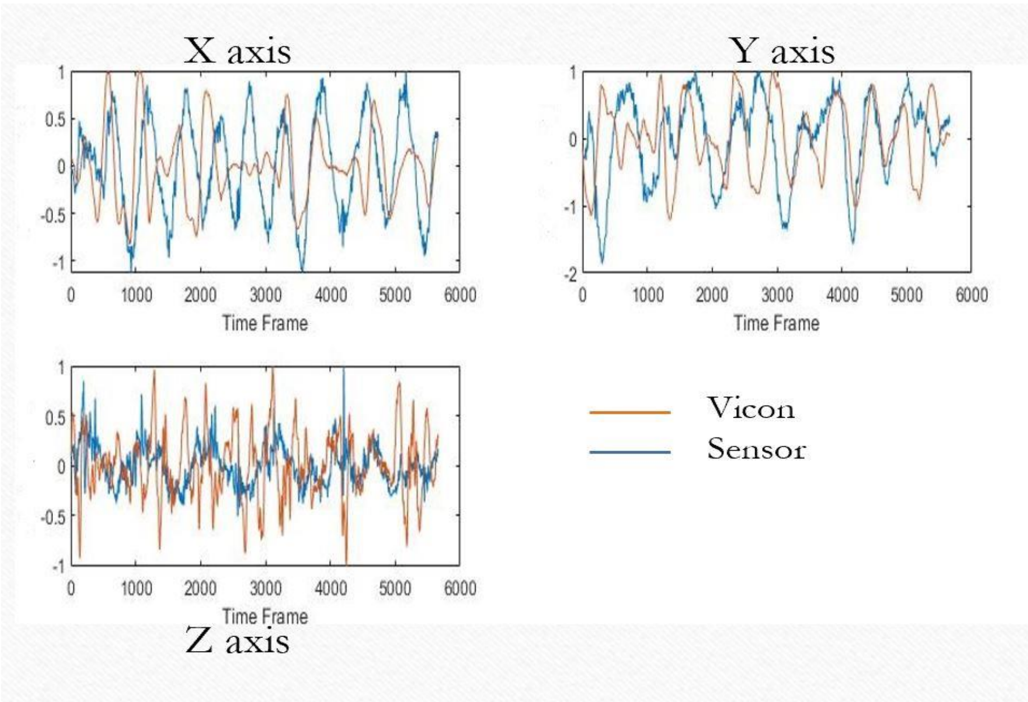

Fig 9. EXG Sway trial 3

CHAPTER IV: DISCUSSION

The construction of the IMU sensor hardware to collect and record the accelerometer data of the sensor LSM9DS1 to measure acceleration is the main goal of the project. The study was successful enabling the constructed IMU to collect the data. The study also focused on the comparison results of the IMU sensor with the motion capture system. The results varied across trials from strong to weak with, a very weak correlation on Z-axis throughout the trial.

Although the data from the sensor is collected, there was difficulty in recording the data from the Arduino, from setting the baud rate, since it determines the transmission rate between the computer and the Arduino. For some unknown reasons the data recorded from the Arduino, has several data errors, and there were missing data points. Apart from using python code, we have also tried other software to collect the data from the Arduino, regardless there were data errors, and comparatively, the customized python code was the most reliable among them. We had trouble coding the accelerometer to have a constant frequency throughout all the trials. the default sampling frequency of the sensor is 119 Hz [22,23], however, we have collected the data at a varying rate of 191 - 192 Hz. The sensor sampling rate was fluctuating for every trial, which makes the sampling of data unreliable. The change in the sampling rate may be due to the triple transmission of data from the sensor. The data is transmitted from the sensor followed by the Arduino and then the python code, this transmission might decrease the sampling rate. The other applications to record data such as "coolterm" and "teraterm" (other software used record the data from the Arduino), variation in frequency. Error accumulation is a common possible error in accelerometer data. These error accumulations may be due to the continuous collection of data, while the sensor is powered using the USB cable. One way to overcome this may be changing the sensor to an accelerometer and placed in multiple parts of the body preferably the sternum and lumbar [27-29,15-19] and then using a sensor fusion algorithm [30].

After fusing if we still have some missing points or random values in the sensor, we can use Kalman Filter [30], which can help identify the nearest value of the sensor.

The anticipated results of the comparison of the motion capture and the IMU sensor are to be similar, however results varied across trials. From Table 1 we can see the correlation of the normalized data points to vary among the trials and ranges from weak, moderate, and strong. However, the Z-axis has a weak correlation among all the others. The correlation value of the pre-processed data was also calculated, and it shows there was an improvement, overall, among the coordinates of the trials. Of these, data processing steps appeared to have the least influence on Z-axis correlation values. The plotted graphs show us that there were some similar signal patterns with some phase shifts among the trials with the exclusion of the Z-axis.

Correlation results for the Z axis may be explained by the longitudinal axis where movement is done rarely, or by the internal fictitious force experienced by the sensor while moving whereas the accelerometer measures the position and velocity based on the pressure exerted in the inner axis of the sensor[33]. The sampling rate of the motion capture system is 200 Hz, whereas the sensor is approximately 192 Hz. Therefore, the data is resampled and normalized. The data processing might affect the original data collected from the sensor. This can also be done by some other algorithms such as decision tree[31], Principal component analysis(PCA) [32], however, these machine learning algorithms are complicated in process and have their disadvantages, henceforth, it is still valuable to perform normalization. However, when this sensor is to be used in the outdoor or sports it might have good use since we are not concerned about the sampling frequency of the sensor. This requires further research on how to control the sampling frequency of the sensor, as well as to collect the data without errors. Further research is required to compare the data, or probably designing the sensor with a new algorithm.

CHAPTER V: CONCLUSION

The current study shows that the successful construction of the sensor has data errors and the comparison among the sensor and the Vicon varies among the trials for pre and post data processing from weak to strong. These days accelerometers are cost-effective, readily available sensors, and can be embedded in many devices, such as smartphones, activity trackers that include Virtual Reality as well. They are easy to use and non-invasive for the individuals. Since the accelerometers are easy to coalesce, the virtual reality feedback with the help of a sensor should be feasible. The importance of visual biofeedback as already explained can give effective results in the field of training and rehabilitation, which can be a future work of this project.

REFERENCES

1. 1. Ekdahl C, Jarnlo GB, Andersson SI. Standing balance in healthy subjects. Evaluation of a quantitative test battery on a force platform. Scand J Rehabil Med. 1989;21(4):187-195.
2. https://apps.dtic.mil/sti/pdfs/ADA425327.pdf
3. T. Sakaguchi, T. Kanamori, H. Katayose, K. Sato and S. Inokuchi, Human motion capture by integrating gyroscopes and accelerometers, 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems (Cat. No.96TH8242), Washington, DC, USA, 1996, pp. 470-475, DOI: 10.1109/MFI.1996.572219.
4. Giggins, O.M., Persson, U.M. & Caulfield, B. Biofeedback in rehabilitation. J NeuroEngineering Rehabil **10,** 60 (2013). https://doi.org/10.1186/1743-0003-10-60
5. Alba, Alejandro. "MIT researchers create a device that can recognize, track people through walls". nydailynews.com. Retrieved 2019-12-09.
6. https://simplifaster.com/articles/3d-motion-capture-sport/
7. https://cdn-learn.adafruit.com/
8. Seel, Thomas; Raisch, Jörg; Schauer, Thomas. 2014. IMU-Based Joint Angle Measurement for Gait Analysis. Sensors 14, no. 4: 6891-6909.
9. F. Eric, Inertial head-tracker sensor fusion by a complementary separate-bias Kalman filter, in Proc. VRAIS ’96, Santa Clara, CA, 30 March - 3 April 1996, pp. 185-194
10. J. Eriksson, M. J. Matari´c, and C. J. Winstein, Hands-off assistive robotics for post-stroke arm rehabilitation," in Proc. IEEE 9th Int. Conf. Rehabilitation Robotics, Chicago, USA, June 28 – July 1, 2005, pp. 21-24.

11. Z. Q. Ding, Z. Q. Luo, A. Caruso, I. M. Chen, et al., Inertia sensor-based guidance system for upper-limb posture correction, in Medical Engineering & Physics, vol. 35, pp. 269-276, 2013.
12. W. W. Wang and L. C. Fu, Mirror therapy with an exoskeleton upper-limb robot based on the IMU measurement system, in IEEE International Workshop on Medical Measurements and Applications Proceedings (MeMeA), Bari, May 30-31, 2011, pp. 370-375.
13. Oancea a.m., sararu t.a. 2016 Arduino based calibration of an inertial sensor in view of a gnss/imu integration". Scientific Papers. Series "Journal of Young Scientist, Vol. 4, PRINT ISSN 2284-8027, 177-182.
14. L. Tran, Data Fusion with 9 degrees of freedom Inertial Measurement Unit to determine object's orientation, Senior Project, June 2017.
15. Abe Y, Sakamoto M, Nakazawa R, Shirakura K. Relationship between joint motion and acceleration during single-leg standing in healthy male adults. J Phys Ther Sci. 2015;27(4):1251–6.
16. Han S, Lee D, Lee S. A study on the reliability of measuring dynamic balance ability using a smartphone. J Phys Ther Sci.2016;28(9):2515–8.
17. Heebner NR, Akins JS, Lephart SM, Sell TC. Reliability and validity of an accelerometry based measure of static and dynamic postural stability in healthy and active individuals. Gait Posture. 2015;41(2):535–9.
18. Neville C, Ludlow C, Rieger B. Measuring postural stability with an inertial sensor: validity and sensitivity. Med Devices (Auckl). 2015; 8:447–55.
19. Rouis A, Rezzoug N, Gorce P. Validity of a low-cost wearable device for body sway parameter evaluation. Comput Methods Biomech Biomed Eng. 2014;17(Suppl 1):182–183.

20. Wilbert J.Lick mentioned in Technology and Engineering, 2012.
21. www.adafruit.com
22. www.arduino.cc/forum/
23. Horak FB. Clinical assessment of balance disorders. Gait Posture. 1997; 6:74-86.
24. https://doi.org/10.1016/S0021-9290(01)00231-7
25. http://www.statstutor.ac.uk/resources/uploaded/pearsons.pdf
26. Seimetz C, Tan D, Katayama R, Lockhart T. A comparison between methods of measuring postrual stability: force plates versus accelerometers. Biomed Sci Instrum. 2012; 48:386–92.
27. Dabbs NC, Sauls NM, Zayer A, Chander H. Balance performance in collegiate athletes: a comparison of balance error scoring system measures. J Funct Morphol Kinesiol. 2017;2(3):26.
28. Amick RZ, Chaparro A, Patterson JA, Jorgensen MJ. Test-retest reliability of the sway balance mobile application. J MTM. 2015;4(2):40–7.
29. Kooij, H.v.d., et al., A multisensory integration model of human stance control. Biological Cybernetics, 1999. 80: p. 299-308.
30. https://math.stackexchange.com/questions/927190/normalization-of-data-in-decision-tree
31. https://chanzuckerberg.github.io/scRNA-python-workshop/preprocessing/02-normalization.html
32. http://www.starlino.com/imu_guide.html
33. Siciliano, Bruno; Khatib, Oussama (20 May 2008). Springer Handbook of Robotics. Springer Science & Business Media. ISBN 9783540239574 – via Google Books.

APPENDIX A: ARDUINO SCRIPT

```
#include "Arduino_LSM9DS1.h"
void setup() {
   Serial.begin(115200);
   while(!Serial);         //Wait for serial connection
   delay(10);
   if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1); }
  IMU.accelUnit=METERPERSECOND2;

  // Link to the source code. https://github.com/FemmeVerbeek/Arduino_LSM9DS1
  // Check the above link (Section 4) for setting the desired frequency
   IMU.setAccelODR(5); // Value 4 - 238Hz for Accel & Gyro; 10Hz for Magno
  //IMU.setAccelODR(3); // Value 3 - 119Hz for Accel & Gyro; 5Hz for Magno

  // Calibration values
  IMU.accelOffset[0] = -0.031748;   IMU.accelOffset[1] = -0.013127;   IMU.accelOffset[2] =
-0.005296;
  IMU.accelSlope [0] = 0.992983;   IMU.accelSlope [1] = 0.994812;   IMU.accelSlope [2] =
0.998038;
  IMU.gyroOffset[0] = 0.846509;   IMU.gyroOffset[1] = 1.105225;   IMU.gyroOffset[2] = -
0.271899;
  IMU.gyroSlope   [0] = 1.290758;   IMU.gyroSlope   [1] = 1.150051;   IMU.gyroSlope   [2] =
1.223229;
  IMU.magnetOffset[0] = 25.492554;   IMU.magnetOffset[1] = 14.613647;
IMU.magnetOffset[2] = 4.839478;
  IMU.magnetSlope [0] = 1.166290;   IMU.magnetSlope [1] = 1.137117;   IMU.magnetSlope
[2] = 1.118424;


}

void loop() {
  // Acceleration
  float uncalAX, uncalAY, uncalAZ;
  float calAX, calAY, calAZ;

  IMU.readRawAccel(uncalAX, uncalAY, uncalAZ);
  IMU.readAccel(calAX, calAY, calAZ);

  Serial.print(uncalAX);Serial.print(",");
  Serial.print(uncalAY);Serial.print(",");
  Serial.print(uncalAZ);Serial.print(",");
  Serial.print(calAX);Serial.print(",");
  Serial.print(calAY);Serial.print(",");
  Serial.print(calAZ);Serial.print(",");

}
```

APPENDIX B: PYTHON SCRIPT

```
import serial
import csv
import sys
import serial.tools.list_ports
from datetime import datetime

ser = None
connected = False

def checkPorts():
   global connected
   ports = [tuple(p) for p in list(serial.tools.list_ports.comports())]
   #print(ports)
   com3Port = [port for port in ports if 'COM3' in port]
   if len(com3Port) is not 0 and 'COM3' in com3Port[0]:
      connected = True
   else:
      connected = False

def checkSerialConnection():
   global ser
   global connected
   while ser is None:
      checkPorts()
      if connected:
         ser = serial.Serial('COM3', 115200)
         print ("connected")
         ser.flushInput()

def readData():
   global connected
   #IMPORTANT: Change the file name ("data.csv") to different names for different motions.
   with open("sample2.csv", "w", newline='') as f:
      data =""
      while connected: #and datetime.datetime.now() < nextMin:
         try:
            ser_bytes = ser.readline()
            data = str(ser_bytes, 'utf-8')
            ts = datetime.now().strftime('%H:%M:%S')
            data = ts+","+data
            print(data)
            strsplit = data.split(",")

            writer = csv.writer(f, delimiter=",")
            writer.writerow([(x) for x in strsplit])
         except Exception:
            checkPorts()
            print("Error converting - " + data)
            print("Keyboard Interrupt")

if __name__ == "__main__":
   checkSerialConnection()
   readData()
```

APPENDIX C: MATLAB SCRIPT

```
%% Umaiyaal Thesis Code %%
close all;
clear all;
clc;

data = xlsread('EXG S&V_3_192 Hz');
%sens data col 14,15,16; vicon data col 18,19,20
% (possibly) calculate cal sens data multiply col 10,11,12 by 1000 (11 = z; 12 =y)
% remove missing values from sensor data (col 14,15,16)
% filter sensor data using medfilt1 -- for loop
% resample com data to match sensor
% trim data to elimitate Z synch peaks
    % look at Z sensor plot to determine cut points
    % input start and end point indeces to set range
% calculate com acceleration -- pad if necessary
% plot sensor and com accelerations for comparison
%% DATA PROCESSING
%remove 'errant' zeros and NaNs from series
sax = nonzeros(rmmissing(data(:,14)));
say = nonzeros(rmmissing(data(:,15)));
saz = nonzeros(rmmissing(data(:,16)));
%filter sensor using 3rd order median filtering to remove noise
fsax = medfilt1(sax);
fsay = medfilt1(say);
fsaz = medfilt1(saz);
figure(1);
plot(fsax);
figure(2);
plot(fsay);
figure(3);
plot(fsaz);
pause = input('Pause here to remove remaining sensor noise');
%grab com data from spreadsheet
comx = data(2:end,18);
comy = data(2:end,19);
comz = data(2:end,20);
%calculate vicon com accel - pad if necessary
com_ax = medfilt1(gradient(gradient(comx)));
com_ay = medfilt1(gradient(gradient(comy)));
com_az = medfilt1(gradient(gradient(comz)));
%smooth vicon com accel data
scom_ax = smoothdata(com_ax,'movmean',100);
scom_ay = smoothdata(com_ay,'movmean',100);
scom_az = smoothdata(com_az,'movmean',100);
figure(4);
plot(scom_ax);
figure(5);
plot(scom_ay);
figure(6);
plot(scom_az);
pause = input('Pause here to remove com noise');
%check sensor z axis plot to set cut points
plot(fsaz); %check cut points
trim1 = input('first cut frame');
trim2 = input('last cut frame');
%trim sensor series to eliminate start/end indicator spikes
tfsax = fsax(trim1:trim2);
tfsay = fsay(trim1:trim2);
tfsaz = fsaz(trim1:trim2);
```

```
%determine cuts for com data by using trim 1&2 as percent of series
t3 = trim1/length(fsax);
t4 = trim2/length(fsax);
trim3 = round(length(scom_ax)*t3,0);
trim4 = round(length(scom_ax)*t4,0);
%trim vicon series to eliminate start/end indicator spikes
tcomx = scom_ax(trim3:trim4);
tcomy = scom_ay(trim3:trim4);
tcomz = scom_az(trim3:trim4);
%resample vicon data to match sensor series
comx2 = resample(tcomx,length(tfsax),length(tcomx));
comy2 = resample(tcomy,length(tfsay),length(tcomy));
comz2 = resample(tcomz,length(tfsaz),length(tcomz));
%% DATA COMPARISON
vicon = [comx2,comy2,comz2];
sensor = [tfsax,tfsay*-1,tfsaz];
%normalize sensor data to surround mean (~zero)
N = size(sensor,1);
M = size(sensor,2);
sensmean = mean(sensor);
for ii = 1:M
    for jj = 1:N
    sensnorm = sensor(jj,ii)- sensmean(:,ii);
    Sensnorm(jj,ii) = sensnorm(:,:);
    end
end
output1 = [Sensnorm,vicon];
%absolute value of vicon and norm sensor data
output_abs = abs(output1);
%scale data to maximum
output1_max = max(output1);
O = size(output1,1);
P = size(output1,2);
for kk = 1:P
    for ll = 1:O
    scale = output1(ll,kk)/output1_max(:,kk);
    output2(ll,kk) = scale(:,:);
    end
end


figure(7);
plot(output2(:,1));
hold on
plot(output2(:,4));

figure(8);
plot(output2(:,2));
hold on
plot(output2(:,5));

figure(9);
plot(output2(:,3));
hold on
plot(output2(:,6));

xlswrite('EXG_3_output',output2);
%
% [locx,xI] = max(sens_ax);
% [locy,yI] = max(sens_ay);
% [locz,zI] = max(sens_az);
```