2013

Recreating the Trajectory of a Golf Swing Using a Microelectromechanical System

Blagoy Yordanov Kaloferov
Bard College

Recreating the Trajectory of a Golf Swing Using a Microelectromechanical System

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Blagoy Kaloferov

Annandale-on-Hudson, New York
May, 2013

Abstract

In the game of golf, practicing and improving the swing is a fundamental part of learning. Unfortunately, analyzing one's swing and all of its features is a costly and difficult task as it involves high speed cameras or the assistance of a coach. This paper presents a novel approach to golf swing analysis using a microelectromechanical system containing sensors that are attached to a golf club. This novel hardware system has been constructed using commercially available components. A displacement estimation algorithm compensates for the errors of the MEMS sensors. The raw data from the sensors is processed using a sensor fusion orientation estimation algorithm and the displacement estimation algorithm. Finally an application has been developed to help visualize the trajectory of the golf swing and give feedback on the swing's key features. The hardware prototype built for the purpose has 9 degrees of freedom, communicates wirelessly and has expandable storage. Because it is both an attitude and heading reference system and an inertial measurement unit its use could be extended to other sports or applications.

Contents

List of Figures

Dedication

To my family for their love and support.

Acknowledgments

First of all, I would like to thank my advisor Rebecca Thomas for giving me the right directions when necessary, but also letting me explore on my own, all possible choices regarding this project. That gave me a perfect balance of learning new things, and progressing adequately with the project, throughout the past year.

BIG thanks goes to my family for always being there for me and guiding me through times of difficult choices. I thank my girlfriend Angela, for always supporting me and motivating me to work hard and be passionate for the things that I do.

Thanks to Seth Tuckerman for helping me come up with the idea for this project, and showing me how my work could be relevant to a golf player.

Finally thanks to my friends at the computer science department Anis Zaman and Nabil Hossain, without whom my experience at Bard would not have been the same.

1 Introduction

More than 29 million people in the USA play golf, which involves the use of clubs to hit a small ball into series of holes. For a successful golf game, huge importance is placed on the golf swing. While there is sufficient literature written on the subject, an amateur golfer would benefit enormously from having instant feedback on his or her swing. A beginner golfer would like to recognize his or her mistakes and try to correct them. Usually there are two options for that purpose: a coach can look at the club during a golf swing and use accumulated experience to judge how good it is. Alternatively, a high speed camera could record a video of the swing. Both options could be very useful, but come at a high price. The high-speed camera requires calibration and is very expensive, not quite fitting the role of a device that one could set up and use anywhere. Having an instructor assist a learning golfer is viable, but does not promise technical accuracy and is often economically unjustified for the average golfer. This senior project aims at the development of hardware and software that could achieve a good balance between these properties: easy set up, practical functionality, accurate feedback and low price.

The device I designed has an accelerometer, a gyroscope and a magnetometer as sensors attached on a common board. An accelerometer measures the gravitational force and can be found in almost every modern smartphone. A gyroscope measures rotational velocity and a magnetometer measures the strength of earths' magnetic fields. The device also has a microcontroller that controls and utilizes the sensors. The system is powered by a battery and has sufficient memory to store a day's worth of swings. A microelectromechanical accelerometer, gyroscope and magnetometer are used to acquire raw data that is filtered and processed to obtain the displacement and orientation of a golf clubhead during a golf swing. The prototype system build for this project has 9 degrees of freedom, expandable storage capacity and Bluetooth module. The final device has to be physically attached to the club.

Some of the main challenges to this project come from the nature of the sensors. Since they are not military grade sensors, they are prone to small errors that could alter the output of the processing algorithms. During a golf swing, the rotational velocity of the golf club could reach 1500 degrees per second and the g force that the clubhead experiences can be as high as $16g$. The time frame between the beginning of the down-swing motion and the impact of the ball is roughly $\frac{1}{3}$ of a second, which implies the need for a very high sensor sampling rate and impeccable accuracy. [9] The device's performance, as shown in this paper, demonstrates that although constructed using cheap commercial components, it can serve as a reliable inertial measurement unit (IMU) and attitude and heading reference system (AHRS). This paper proposes a methodology for capturing and analyzing the following features of a golf swing: the golf club's trajectory, velocity and its orientation throughout the golf swing. In addition to that, we are proposing a software application that interfaces the prototype system.

The rest of the paper is organized as follows. Chapter 2 provides an introduction to all components chosen for the system. It also gives an overview of the design of the hardware prototype that was built. Chapter 3 explains the theory behind the algorithm employed for orientation sensing and gravity compensation and how displacement should be calculated to minimize errors. Chapter 4 explains which features of a golf swing can be captured by the proposed device and how they are relevant to a golfer. It also examines an application, designed to facilitate the hardware system, attached to a golf club. Chapter 5 provides techniques for optimizing the performance of the system and an empirical evaluation of the efficiency of these techniques. Chapter 6 presents a golf swing simulation equipment, used to evaluate the performance of the device and its processing algorithms. The paper closes with conclusion and future work suggestions in Chapter 7.

2
MEMS Sensors and Prototype Hardware Configuration

Microelectromechanical sensors are miniature devices that convert a non-electrical quantity into an electrical signal. By integrating a micro sensor with a microprocessor, a smart sensor called a microelectromechanical system (MEMS) is produced. MEMS systems contain electrical and mechanical parts. [8] Accelerometers, gyroscopes and magnetometers are MEMS sensors that the prototype system described in this paper incorporates. They convert energy from one form to another, thus generating output. When aligned together on a common board the sensors can be used to perform complicated tasks. Having the sensors aligned means that their x, y and z axis are parallel to each other. The prototype system also includes other components, forming an inertial measurement unit (IMU) which is composed of sensors capable of measuring proper acceleration and angular velocity. For simplicity, this paper will refer to the prototype system described below simply as an IMU.

It is difficult to find commercially available hardware products that are portable, act as an IMU and store and transfer data wirelessly. Thus, from the beginning of this project, I was motivated by the idea of designing a device that could be assembled using only components that are easily available on the electronics market. For such an IMU configuration the following constraints need to be considered: price, portability, convenient power source, storage capacity, durability and accuracy. This chapter provides a description of the mechanics of the MEMS sensors chosen for the project and the other pieces of hardware that make the prototype work as desired.

2.1 Accelerometer

2.1.1 Mechanical Accelerometer

An accelerometer is a device that will measure acceleration forces. These forces may be static, like the force of gravity, or they could be dynamic, caused by moving or vibrating the accelerometer. A completely stationary accelerometer on the earth's surface would measure a force of $1g$. If an accelerometer is stationary in space it will have output of $0g$ in each of its axes. One can think of accelerometers as systems that have a mass attached to a spring. Figure 2.1.1. shows how a basic, one-axis accelerometer is modeled. Using Newton's law, $F = m * A$, where m is the mass and A is the acceleration, one can calculate the acceleration of the sensor.

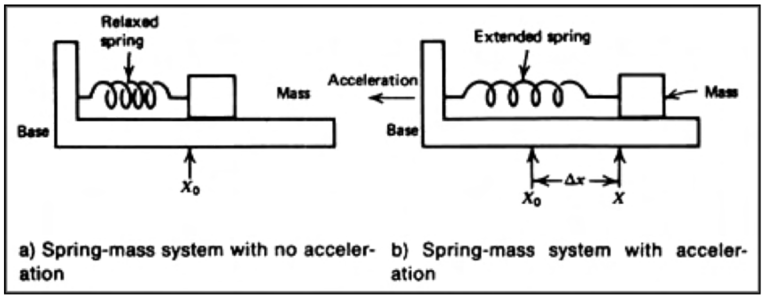

Relaxed spring
Mass
Base
X_0
Acceleration
Extended spring
Mass
Base
X_0
Δx
X
a) Spring-mass system with no acceleration
b) Spring-mass system with acceleration

Figure 2.1.1: Basic spring-mass system accelerometer
[21]

2.1.2 MEMS Accelerometer

A MEMS accelerometer has many advantages compared to mechanical accelerometers. These advantages include low cost, better performance, compactness, and multi axis sensing on a silicon structure. MEMS accelerometers are made of a micro machined structure that is mounted on a silicon base. It has a spring with mass-displacer that can translate external forces such gravity into kinetic motion energy. (figure 2.1.2) As that mass experiences external forces due to gravity or displacement the capacitor sensors that surround the mass reflect those forces by outputting a change of voltage. [4] Low cost MEMS accelerometers are used in many devices nowadays and are therefore cheap. They are used for orientation sensing in smartphones, free fall detection and even impact detection in automobile airbags.

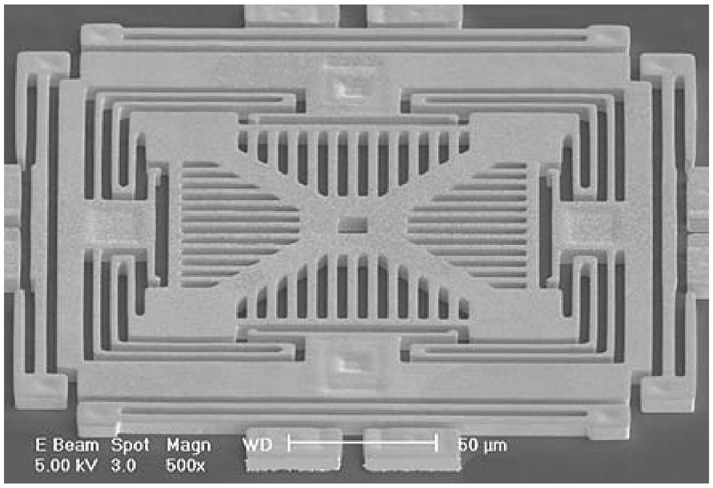


Figure 2.1.2: The core micromechanical structure of a 3-axis MEMS accelerometer [7]

2.1.3 MEMS Accelerometer Error Sources

MEMS accelerometers have lower power consumption and faster start-up times compared to mechanical accelerometers. However, the sensors show some errors in their measurements including dynamic error, such as measurement noise, and static errors, such as the offset difference between the acceleration measured by the accelerometer and the actual acceleration. Temperature changes can disturb accelerometer output too. [32]

2.1.4 Accelerometer ADXL345

For this project I chose to use accelerometer ADXL345 manufactured by Analog Devices. The ADXL345 is a small, thin, low-power, 3-axis accelerometer with high resolution (13-bit) measurement at up to 16 g. Digital output data is accessible through I^2C digital interface. I^2C is a two wire interface allowing for communication between a microcontroller and peripheral device. In addition to that this model has been used in a variety of devices and due to the high volumes in production it is relatively cheap. [1]

2.2 Gyroscope

A gyroscope is a device that is used for orientation sensing based on angular momentum. In its basic mechanical form it is a spinning wheel that is free to assume any orientation. The wheel is attached on two gimbals and since it is rotating rapidly, its momentum keeps its orientation fixed when the gimbals' orientations are changed. The gyroscope measures the angle between the adjacent gimbals to estimate orientation. [32] Mechanical gyroscopes are accurate, but are expensive, require moving parts and cannot be constructed on micro scale.

2.2.1 MEMS Gyroscope

MEMS gyroscopes are made with the use of silicon micro-machining techniques. They are small and can be very inexpensive since MEMS gyroscopes are widely used in smartphones and game controllers. MEMS gyroscopes contain a vibrating structure that because of its vibration remains unaffected by the orientation change of its support. MEMS gyroscopes are excellent candidates for the data recording device of this project as they have small weight and size, low price and low power consumption.

2.2.2 MEMS Gyroscope Sources of Error

MEMS gyroscopes' output is prone to errors that we must note as they are of importance in this paper. A constant bias error is the average output of a gyro, when it is not undergoing any rotation. Offset error is the difference between the output of the gyro and its true value. The output of gyroscopes is also negatively affected by mechanical noise, which fluctuates at a rate much greater than the sampling rate of the actual sensor. Finally temperature changes in the sensor fluctuate its output. [32]

2.2.3 Gyroscope ITG3200

MEMS gyroscope ITG-3200 is a 3-axis gyroscope produced by Invensense. The chip supports Fast-Mode I^2C (400 kHz) interface and has an embedded temperature sensor. It can detect angular change of up to 2000 degrees per second which makes it capable of accurately recording golf swings that have fast rotational velocity of 1500 degrees per second. [14]

2.3 Magnetometer

A Magnetometer is a type of sensor that measures the strength and direction of its local magnetic field. The magnetic field measured will be a combination of the earth's magnetic field and any magnetic field created by nearby objects. [29] The Earth generates a magnetic field that creates measurable magnetic disturbances in the atmosphere. A magnetometer measures this phenomenon in terms of magnetic flux density. That magnetic flux can be treated as a vector that has direction and magnitude. [6] A magnetometer is an ideal supplement to an inertial measurement unit, because it is inexpensive and has low power consumption.

2.3.1 Magnetometer sources of error

Magnetometers are not perfect because their output is distorted by other magnetic fields, including the magnetic fields created by components near the magnetometer. If magnetometers were completely reliable, we would not need gyroscopes and accelerometers. Magnetometer readings need to be filtered and corrected because of any distortion that they are subject to. [15] An orientation estimation sensor fusion algorithm presented in section 3.2.2 elaborates on such adjustment technique.

2.4 Magnetometer HMC5883L

I chose to use MEMS magnetometer HMC5883L produced by Honeywell. The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing with I^2C digital interface for applications such as low-cost compassing and magnetometry. The HMC5883L can produce 12-bit output at rate of 160 Hz. The sensor is popular for electronics prototyping and therefore has good online community support. [13]

2.5 Wireless Communication

For wireless communication, I was faced with the choice of using a low power radio chip, a Bluetooth module or a Wi-Fi chip. A radio chip has very low power consumption, which is good for a battery-powered device, but it is also limited by the fact that it can only communicate with another radio chip that has the same frequency. That makes pairing the system with smartphones or computers difficult. A Wi-Fi module has the best speed and range but its high battery consumption could negatively affect the performance of the IMU. Bluetooth chips contain the best balance between speed, communication range and battery consumption and most of all, almost all modern mobile devices and computers are Bluetooth enabled. The chip I chose is the Bluetooth Silver Mate, produced and sold by

Sparkfun Electronics. It has FCC Approved Class 2 Bluetooth radio modem that supports serial speed of up to 115200 bps.

2.6 Data Logging

2.6.1 SD Card Storage Expansion

The storage capacity of the microcontroller, used for the IMU device is insufficient to store more than few seconds of data, due to its limited memory capacity. (table 2.7.1) Therefore, some additional storage space for data logging was essential to the success of the project. I decided to use a SD card adaptor that would enable me to store a practically unlimited amount of data. The drivers for the adapter are incorporated in microcontroller library called SDFat. SDFat is an Arduino library that supports FAT16 and FAT32 file systems on standard and high capacity SD cards. [25] I created custom code that would save the data according to my needs. A Micro SD card of 4 GB, for example, not only makes data logging possible, but is also convenient as it allows an alternative data transfer to a computer.

2.7 Microcontroller Unit

A microcontroller is necessary to integrate all components of the IMU system and to facilitate communication with a computer. A microcontroller is a small computer with integrated processor, memory and input/output peripherals. I decided to work with the Arduino electronics platform and after comparing microcontrollers, I chose a chip that is Arduino compatible but manufactured by a different company, due to its better characteristics.

2.7.1 Arduino electronics and programing platform

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. [2] It is an excellent choice as a microcontroller platform because it has a functional and easy to use IDE, excellent support community and low price. There are various libraries written to support additional components that Arduino microcontrollers can utilize. Some of these components include motors, Wi-Fi, Ethernet shields and even touch-screens. Most of all, there are drivers for dozens of MEMS sensors that were written for Arduino. Initially, I chose to use the most popular Arduino board to date: Arduino UNO. Arduino UNO has all the standard capabilities in the Arduino microcontrollers family and has a good balance between functionality and size. Its price is \$35. Eventually I replaced Arduino UNO with a more suitable microcontroller called Teensy 3.0.

2.7.2 Teensy 3.0

Midway through my senior project at the end of November 2012 a 32-bit microcontroller called Teensy 3.0 was commercially released at the very low price of \$19. It is produced by a small company called PJRC and is now the default choice for many people working on electronics projects because of its specifications. In addition, it is compatible with the Arduino IDE. Teensy is faster, smaller and has better technical specifications as shown in table 2.7.1. [28]

Specifications	Arduino Uno	Teensy 3.0
Processor	Atmega328 16 MHz	MK20DX128 32-bit ARM 48 MHz
RAM Memory	2 KB	16 KB
Flash Memory	32 KB	130 KB
Serial, I^2C, SPI	1, 1, 1	3, 1 ,1
EEPROM	1 KB	2 KB
I/O	14 / 5V	34 / 3.3V
Price	\$35	\$19

Table 2.7.1: Arduino UNO and Teensy 3.0 technical specifications

Some mathematical operations have to be performed on the microcontroller for this project as shown in Chapter 4. Therefore, more RAM and clock speed that is 3 times that of Arduino UNO are important for increasing the sample rate of the IMU system. As a result, I chose to use Teensy 3.0 as the microcontroller for final IMU design and evaluation.

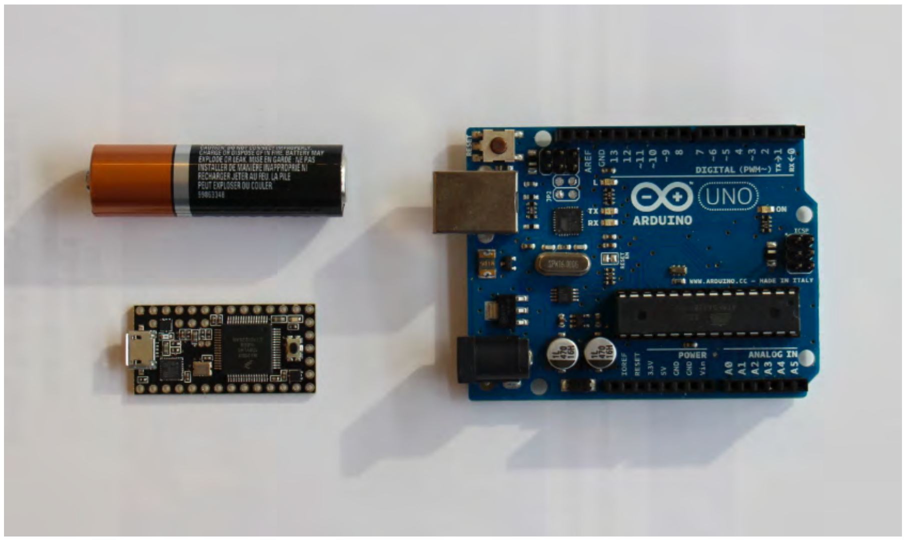


Figure 2.7.1: Arduino Uno, AA battery and Teensy 3.0 size comparison

Figure 2.7.1 shows Teensy's size relative to a Arduino UNO and an AA battery. Teensy 3.0 has the size of one of the most miniature microcontrollers used for electronics prototyping and also has the power and functionality of some of the most expensive microcontrollers on the market up to date. However, I had to optimize all the drivers for the sensors I have chosen due to the drivers' incompatibility with 32-bit microcontrollers. In the rest of the paper, any reference to Arduino compatible code implies that the code is compatible with Teensy 3.0 unless specifically noted otherwise.

2.8 9 Degrees of Freedom Sensor Stick

The three MEMS sensors that I chose to use are incorporated on the same board and sold by SparkFun Electronics. The board is called Sensor Stick and its combination of sensors gives it 9 degrees of freedom. It includes ADXL345 accelerometer, HMC5883L magnetometer, and ITG-3200 gyroscope as introduced earlier in this chapter. The board has I^{2}C interface and is very small, measuring just 34x10 mm. (figure 2.8.1)

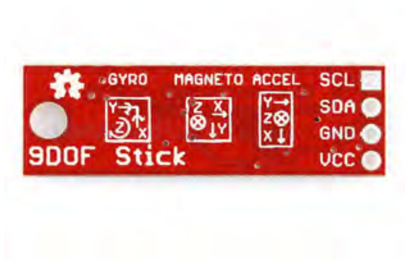


Figure 2.8.1: Sensor Stick sold by SparkFun.
[26]

2.9 Final Prototype Configuration

This section provides an overview of the design of the prototype system, which is physically attached to a golf club and has all components described in this chapter.

2.9.1 Hardware configuration

The final IMU system has a main board that connects and powers all of its elements. Figure 2.9.5 shows the board's layout. In order to make the device adaptive, I avoided soldering components directly to the main board. That is important for a prototype system because its elements can be treated as modules and replaced. The board can be placed in a protective enclosure. The system could be even smaller, should all elements be soldered directly onto a main board or should one choose to use Printed Circuit Board software to design a platform containing all components. These options are beyond the scope of this project but we have provided a list of all elements for such purposes. Figure 2.9.1 provides a picture of all individual elements together. They can be easily purchased from any major online retailer like SparkFun Electronics. The list of the components is as follows:

1. **Teensy 3.0** - The main microcontroller unit. It powers and controls all components.

2. **Sensor Stick** - The sensors board. Having accelerometer, gyroscope and magnetometer means that the sensor board gives the device 9 degrees of freedom. It connects to Teensy 3.0 using I^2C interface.

3. **Micro SD card adapter** - When combined with an MicroSD card the adapter will enable storage expansion. That is important because the data cannot be stored in the microcontroller.

4. **Bluetooth Mate Silver** - The module enables wireless communication with computers or tablets. It is connected to the microcontroller using a serial port.

5. **Meas Vibration Sensor** - A vibration sensor (not shown in figure 2.9.1) capable of detecting actions ranging from a single tap to the impact between a golf club and a golf ball.

6. **3.7V 850 mA Battery** - This small, light battery can power the device for more than 2 hours.

7. **PCB Case** - The protective enclosure of the main board that protects all components. It measures 101x50x25 mm and can be opened for easy access to the prototype elements

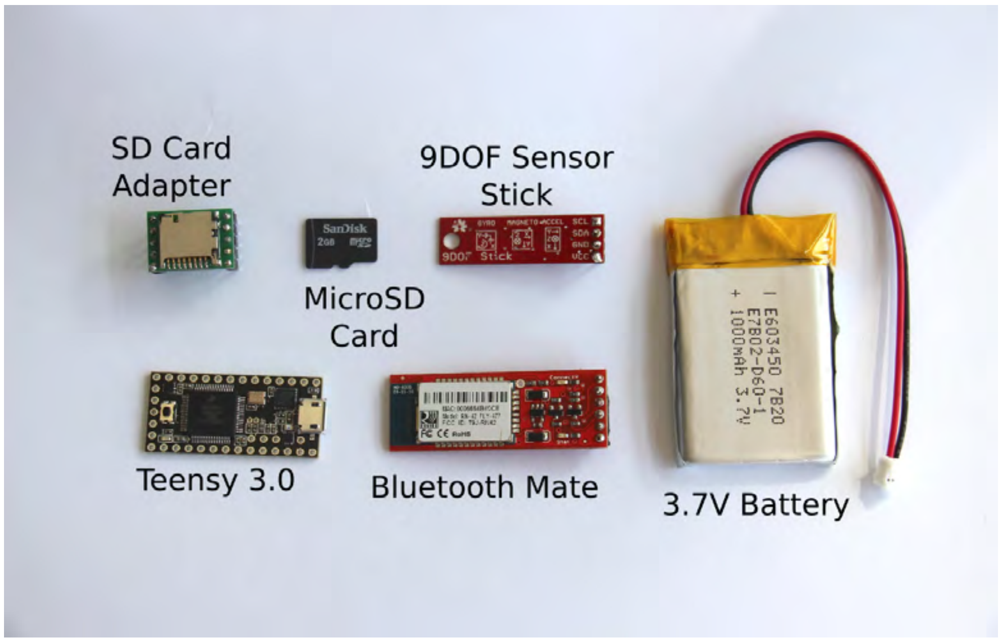


Figure 2.9.1: All major components of the IMU prototype

The IMU can be powered both using its USB connector or a battery and can be controlled over Bluetooth. Sensors communicate with the microcontroller Teensy 3.0 using I^2C bus interface. Additionally, Teensy 3.0 supports up to three hardware serial port connections. We are utilizing one of them for connection with the Bluetooth module which in turn communicates with a receiving computer. Finally, a direct physical connection between Teensy 3.0 and a computer can be established using the microcontroller's built in micro USB port. Figure 2.9.3 shows the assembled device with its enclosure.

Table 2.9.2 shows the price of all components of the hardware configuration. With a total cost of less than $200, the IMU system is cheap considering its multiple features. Furthermore, there are very limited choices when it comes to commercial small MEMS inertial measurement units. Section 6.5.2 provides a comparison between the prototype device and similar commercial product.

Degrees of Freedom	9 DOF
Features	Vibration Sensor, SD card storage expansion
Communication	USB , Bluetooth
Accelerometer	16g
Gyroscope	2000 deg/sec
Size	101x50x25 mm
Sample Rate	400 Hz

Table 2.9.1: Technical specifications of the IMU system

Component	Price
Teensy 3.0	$19
9 DOF Sensor Stick	$99
Bluetooth Mate	$39
SD Card Adapter	$7
850 mA Battery	$10
Meas Vibration Sensor	$3
PCB Enclosure	$5
Other Components	$15
Total Cost	$197

Table 2.9.2: Price of the components of the IMU system

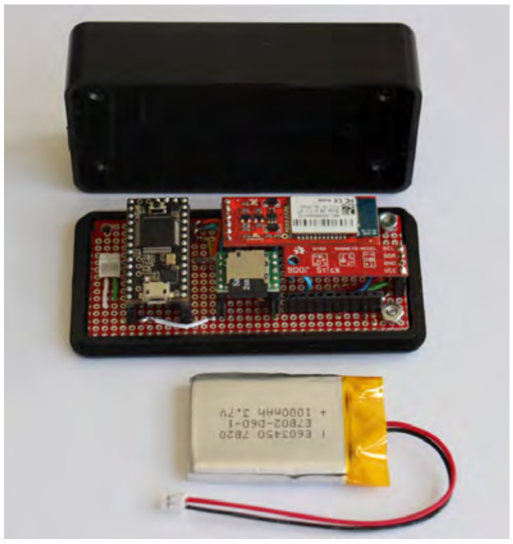


Figure 2.9.2: Top View of the device

Figure 2.9.3: IMU with closed enclosure

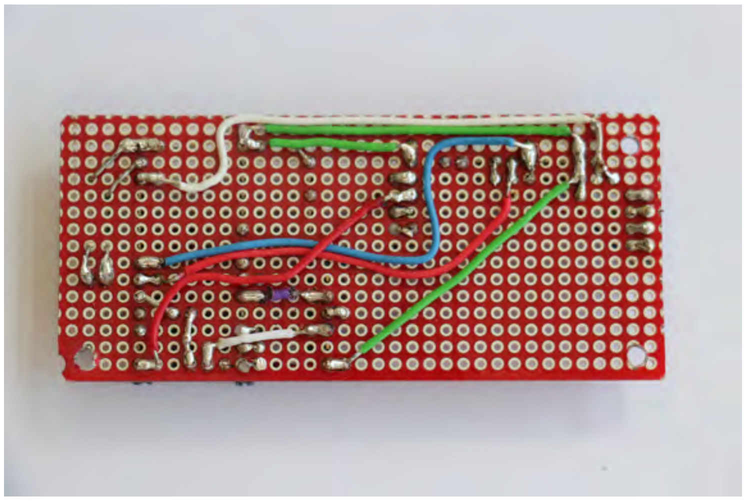


Figure 2.9.4: The back of the IMU main board without the modules

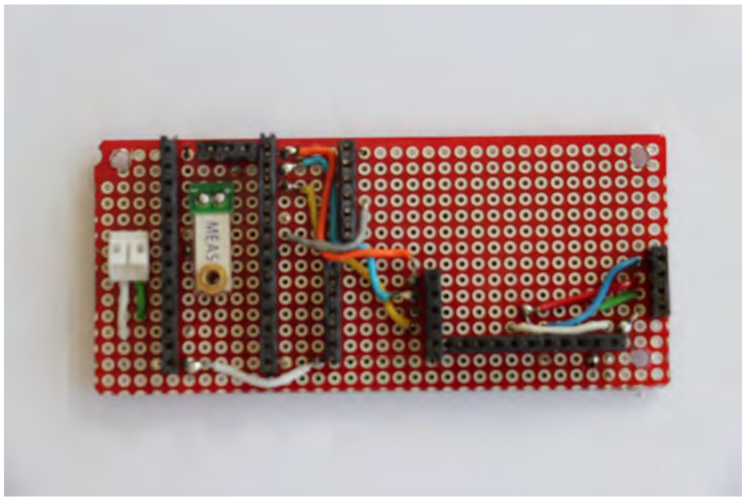


Figure 2.9.5: The front of the IMU main board without the modules

3
Orientation and Displacement Estimation of a Rigid Object

This chapter provides an introduction to orientation sensing using MEMS sensors. Quaternions, which are used to represent orientation are defined and introduced. A magnetic, angular rate, gravity (MARG) sensor fusion algorithm that can utilize a 9 degrees of freedom sensor board is presented. I have adapted this algorithm for the orientation estimation of the IMU device, necessary to determine the clubhead orientation at the moment of impact with a golf ball. The chapter also explains how an object's displacement over time can be calculated using an accelerometer and why the accelerometer's output must be compensated for the earth's gravity.

3.1 Quaternion representation of the orientation of a rigid body

The orientation of a rigid body in three-dimensional space can be represented and stored using quaternions. A quaternion is a four-dimensional complex number that can be treated as a four-dimensional vector, where the last three components represent a direction vector and the first component stands for rotation around that vector.

Quaternions are like extended complex numbers, $w\ +ix\ +jy\ +kz$, where $i^2 = j^2 = k^2 = -1$. This section depicts important properties of quaternions deployed in this chapter in an orientation estimation algorithm. A quaternion ${}^A_B\hat{q}$ represents an orientation of frame B relative to frame A that is achieved through the rotation of angle θ around the the x, y and z axis defined in frame A. Equation 3.1.1 shows how quaternion ${}^A_B\hat{q}$ is defined using unit vector ${}^A\hat{r}$ in the x , y and z axes of frame A respectively. [20]

$$ {}^A_B\hat{q} = [q_1 \quad q_2 \quad q_3 \quad q_4] = [\cos\frac{\theta}{2} \quad -r_x \sin\frac{\theta}{2} \quad -r_y \sin\frac{\theta}{2} \quad -r_z \sin\frac{\theta}{2}] \tag{3.1.1} $$

where r_x, r_y and r_z are the components of the vector ${}^A\hat{r}$.

Relative frames described by an orientation can be swapped using a quaternion conjugate that is denoted by $*$. Equation 3.1.2 shows how the conjugate of a quaternion is obtained. The conjugate of a quaternion ${}^A_B\hat{q}$ could be thought of as ${}^B_A\hat{q}$ as these quaternions describe frame B relative to frame A and frame A relative to frame B respectively.

$$ {}^A_B\hat{q}* = {}^B_A\hat{q} = [q_1 \quad -q_2 \quad -q_3 \quad -q_4] \tag{3.1.2} $$

Quaternion product denoted by $\otimes$, can be used to obtain the compound orientation of two quaternions. (equation 3.1.3). For two quaternions ${}^A_B\hat{q}$ and ${}^B_C\hat{q}$ their compound orientation would be ${}^A_C\hat{q}$.

$$ {}^A_C\hat{q} = {}^B_C\hat{q} \otimes {}^A_B\hat{q} \tag{3.1.3} $$

The product of two quaternions a and b is defined by the Hamilton rule [20] as shown in equation 3.1.4. A quaternion product is not commutative.

$$ a \otimes b = [a_1 \quad a_2 \quad a_3 \quad a_4] \otimes [b_1 \quad b_2 \quad b_3 \quad b_4] \tag{3.1.4} $$

$$ = \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3 \\ a_1b_3 - a_2b_3 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix} \tag{3.1.5} $$

Quaternions can be used to rotate vectors in 3-dimensional space. Using equation 3.1.6 [17] ${}^{A}v$ and ${}^{B}v$ would be the same vectors in sensor frame A and rotated in sensor frame B respectively. The vectors would have to be 4-dimensional with a first element set to 0 for correct quaternion multiplications.

$$^{B}v =^{A}_{B} \hat{q} \otimes^{A} v \otimes^{A}_{B} \hat{q}* \tag{3.1.6}$$

For accurate arithmetic operations quaternions need to be normalized first. They can be easily normalized using equation 3.1.7 [5]

$$^{A}_{B}\hat{q} = [\frac{q_1}{\sqrt{q_1 + q_2 + q_3 + q_4}} \quad \frac{q_2}{\sqrt{q_1 + q_2 + q_3 + q_4}} \quad \frac{q_3}{\sqrt{q_1 + q_2 + q_3 + q_4}} \quad \frac{q_4}{\sqrt{q_1 + q_2 + q_3 + q_4}}] \tag{3.1.7}$$

3.2 Orientation Representation Using Sensor Fusion

To correctly analyze a golf swing the clubs orientation throughout the swing must be estimated. This section describes an algorithm for orientation sensing developed by Sebastian O.H Madgwick. [20]. It estimates orientation based on integrating gyroscope angular rate readings. The algorithm increases the reliability of the orientation sensing by using a magnetometer and an accelerometer to compensate for gyroscope dirft. Sebastian Magwick also incorporated a magnetic distortion compensation algorithm for the magnetometer sensor. The MARG algorithm uses quaternions for orientation representation. Quaternion notations and operations used in this section are presented in section 3.1.

3.2.1 Orientation from Angular Rate Represented using Quaternion

First we will demonstrate how orientation sensing could be achieved using only a gyroscope. Consider a 3-axes gyroscope that measures its angular rate defined as ω_x ,ω_y and

ω_z around its respective x , y and z axis . Having the angular rate in radians for each axis we can construct a quaternion ${}^S\omega$:

$$ {}^S\omega = [0 \quad \omega_x \quad \omega_y \quad \omega_z] \tag{3.2.1} $$

Using the quaternion ${}^S\omega$ we can compute the rate of change of the earth frame relative to the sensor frame denoted by ${}^S_E\dot{q}$:

$$ {}^S_E\dot{q} = \frac{1}{2}{}^S_E\hat{q} \otimes {}^S\omega \tag{3.2.2} $$

Now suppose that we are sampling gyroscope readings ${}^S\omega$ with sample period of Δt. If we are at the n-th sample, then ${}^S\omega_n$ will be the gyroscope readying at sample $n\Delta t$. Let us define ${}^S_E\dot{q}_{e,n}$ to be the orientation of the earth frame relative to the sensor at sample n. Let us assume that we know the initial orientation ${}^S_E\dot{q}_{e,0}$. In that case we can compute the new compound orientation of the device as shown in 3.2.3.

$$ {}^S_E\dot{q}_n = \frac{1}{2}{}^S_E\hat{q}_{e,n-1} \otimes {}^S\omega_n \tag{3.2.3} $$

Finally, we can integrate ${}^S_E\dot{q}_n$ to obtain the estimated orientation of the earth frame ${}^S_E\hat{q}_{e,n}$ relative to the sensors: [20]

$$ {}^S_E\hat{q}_{e,n} = {}^S_E\hat{q}_{e,n-1} + {}^S_E\dot{q}_n\Delta t \tag{3.2.4} $$

The equation above would compute the orientation of an object based only on gyroscope angular rate. As stated in section 2.2.1, MEMS gyroscope readings drift and they cannot correct any orientation estimation errors relative to the earth frame. The combination of a gyroscope with an accelerometer and a magnetometer can compensate for that. Together the three sensors form a magnetic, angular rate, gravity (MARG) sensor array, [20] such as the 9 DOF Sensor Stick used in the prototype device.

3.2.2 MARG Sensor Fusion algorithm

The following presentation is an adaptation of the Arduino implementation of the sensor fusion algorithm by Fabio Verasano. [30] The MARG sensor fusion algorithm estimates gravity and earth's flux vectors using an accelerometer and a magnetometer respectively to compute an adjusted measurement of the gyroscopes angular rate ${}^{S}\omega$. We are going to call that adjusted angular rate ${}^{S}\omega^{est}$.

Consider a 3-axes accelerometer that measures gravitational force a_x, a_y and a_z in its respective axes. We can construct a quaternion ${}^{S}\hat{a}_n$ that contains the gravitational force measurements at sample n:

$$ {}^{S}\hat{a}_n = [0 \quad a_x \quad a_y \quad a_z] \tag{3.2.5}$$

In a similar fashion, we can construct a quaternion that contains a magnetometers' projection of the earth's flux vector at sample n.

$$ {}^{S}\hat{m}_n = [0 \quad a_x \quad a_y \quad a_z] \tag{3.2.6}$$

Together, the following quaternions will be the inputs the the MARG sensor fusion algorithm at sample n:

1. ${}^{S}\hat{a}_n$ - quaternion containing the raw accelerometer readings in the sensor frame.
2. ${}^{S}\hat{m}_n$ - quaternion with magnetometer flux vector in the sensor frame.
3. ${}^{S}\hat{\omega}_n$ - quaternion containing gyroscope angular rate readings in the sensor frame.

The inputs will be used to determine the gravity vector ${}^{S}\hat{v}_{n-1}$ and the estimated direction of the magnetic field in the sensors frame ${}^{S}\hat{b}_{n-1}$ from the previous sample $n-1$. We are concerned with the these vectors because the are used to estimate the difference between the estimated orientation and the true orientation of the sensor. The output of the algorithm is quaternion ${}^{S}_{E}q_{e,n}$ that determines the orientation of the earth frame relative to the sensor. ${}^{S}_{E}q_{e,n}$ is computed as demonstrated in equation 3.2.4 but using the adjusted angular rate ${}^{S}\omega_n^{est}$ instead of the unfiltered angular rate ${}^{S}\omega_n$

3.2.3 Algorithm step

If we assume that we have completed step $n-1$ of the algorithm then we can compute the estimated gravity vector ${}^{S}\hat{v}_{n-1}$ as shown in equation 3.2.7. Note that as explained in equation 3.1.6, if we have a quaternion and its conjugate we can rotate a vector using that quaternion. That is how the gravity projection vector is computed:

$$
{}^{S}\hat{v}_{n-1} = {}^{E}_{S}\hat{q}_{e,n-1} \otimes {}^{E}\hat{v} \otimes {}^{E}_{S}\hat{q}*_{e,n-1} \tag{3.2.7}
$$

If we know the magnetic field vector of the earth ${}^{E}\hat{b}$ we can compute the estimated direction of the magnetic field in the sensor frame:

$$
{}^{S}\hat{b}_n - 1 = {}^{E}_{S}\hat{q}_{e,n-1} \otimes {}^{E}\hat{b} \otimes {}^{E}_{S}\hat{q}*_{e,n-1} \tag{3.2.8}
$$

After that we can compute the error e_n which is the difference between the estimated orientation and the true orientation. That is obtained by summing the cross products of the reference direction of fields and the direction measured by the sensors.

$$
e_n = {}^{S}a_n \times {}^{S}v_{n-1} + {}^{S}m_n \times {}^{S}b_{n-1} \tag{3.2.9}
$$

We sum errors from each step by integrating e_n where k_i is integral gain for rate of convergence of gyroscope biases:

$$
S(e_n) = S(e_{n-1}) + e_n.k_i \tag{3.2.10}
$$

Finally we compute corrected estimated orientation of the angular rate at sample n, ${}^{S}\omega_n^{est}$ as follows:

$$
{}^{S}\omega_n^{est} = {}^{S}\omega_n + e_n.k_p + S(e_n) \tag{3.2.11}
$$

where k_p is the proportional gain that governs convergence to accelerometer and magnetometer readings. Now that we have the adjusted ${}^S\omega_n^{set}$ angular rate, it will be new input for our initial algorithm:

$$ {}_E^S\hat{q}_{e,n} = {}_E^S\hat{q}_{e,n-1} + (\frac{1}{2} {}_E^S\hat{q}_{e,n-1} \otimes {}^S\omega_n^{est}).\Delta t \tag{3.2.12}$$

3.2.4 Magnetic distortion compensation

As shown in equation 3.2.8 we are assuming that we know the magnetic field vector ${}^E\hat{b}$ that is used to compute error e_n. However, magnetometer are prone to distortion of nearby local magnetic fields. Consider that we have measured the direction of the earth's magnetic filed at sample n. That direction is denoted by ${}^E\hat{h}_n$ and can be computed as we rotate the normalized magnetometer measurement ${}^S\hat{m}_n$ by the orientation estimation quaternion ${}_E^S\hat{q}_{e,n-1}$:

$$ {}^E\hat{h}_n = [0\ h_x\ h_y\ h_z] = {}_E^S\hat{q}_{e,n-1} \otimes {}^S\hat{m}_n \otimes {}_E^S\hat{q}_{e,n-1} \tag{3.2.13}$$

The effect of the erroneous inclination of the measured direction earth's magnetic field, ${}^E\hat{h}_n$, can be corrected if the algorithm's reference direction of earth's magnetic field ${}^E\hat{b}_n$ is of the same inclination. That can be achieved by having ${}^E\hat{h}_n$ and ${}^E\hat{b}_n$ normalized to have only components in the earth frame x and z axes:

$$ {}^E\hat{b}_n = [0\ \sqrt{h_x^2 + h_y^2}\ 0\ h_z] \tag{3.2.14}$$

Compensating for magnetic distortions this way ensures that the magnetic disturbances will only affect the estimated heading component of the orientation. [20]

3.2.5 Algorithm pseudo code

A pseudo code of the MARG sensor fusion is presented in algorithm 1. The algorithm takes ${}^{S}a_n$, ${}^{S}m_n$, ${}^{S}\omega_n$, $S(e_n - 1)$ and the quaternion orientation ${}^{S}_{E}\hat{q}_{e,n-1}$ from sample $n - 1$ as inputs. The output is the new quaternion orientation of the earth frame relative to the sensors, ${}^{S}_{E}\hat{q}_{e,n}$. Algorithm 2 and Algorithm 3 are pseudo code representation of supporting methods for the sensor fusion algorithm. Algorithm 2 takes as input the acceleration reading ${}^{S}a_n$ and computes the difference in the estimated direction and measured direction of field vectors. Algorithm 2 filters the magnetometer input ${}^{S}m_n$ using the methodology shown in section 3.2.4 and computes the difference in the expected and measured magnetic direction.

Algorithm 1 AHRS Update

```
1: $Kp$ {proportional gain constant}
2: $Ki$ {integral gain constant}
Require: $Input : g[], a[], m[], q[], integralFB[], deltaT$
Ensure: $qNew[]$
3: $e[]$ {define 3 dimensional array containing the error $e_n$}
4: $qNew[]$ {define a new array that is the output of the algorithm}
5: if isValid(m) then
5:    { Compute feedback from magnetometer measurements. Method isValid() returns
      true if all elements in an array are not equal to 0 }
6:    $m[] \leftarrow normalize(m[])$ {normalize magnetometer readings}
7:    $e[] \leftarrow magnetometerError(m[], q[])$ { Error is sum of cross product between esti-
      mated direction and measured direction of field vectors as shown in equation 3.2.9}
8: end if
9: if isValid(a[]) then
9:    { Compute feedback from accelerometer measurements}
10:   $a[] \leftarrow normalize(a[])$ {normalize accelerometer readings}
11:   $e[] \leftarrow e[] + accelerometerError(a[], q[])$
12: end if
   {Apply feedback only when valid error data has been gathered from the accelerometer
   or magnetometer }
13: if isValid(e[]) then
13:   { Compute and apply integral feedback $S(e_n)$, where integral error $e_n$ is scaled by
      constant Ki for the three axis as shown in 3.2.10}
14:   for $i = 0$ to 2 do
15:      $integralFB[i] \leftarrow integralFB[i] + Ki * e[i] * deltaT$
16:   end for
17:   for $i = 0$ to 2 do
18:      $g[i] \leftarrow g[i] + integralFB[i] + Kp * e[i]$ {apply the updated integral feedback and
         proportional feedback for the three axis of the gyroscope readings}
19:   end for
20: end if
   { Finally integrate rate of change of quaternion as shown in equation 3.2.12}
21: for $i = 0$ to 2 do
22:   $g[0] \leftarrow g[0] * (0.5f * deltaT)$ {pre-multiply common factors }
23: end for
24: $qNew[0] \leftarrow q[0] + (-q[1] * g[0] - q[2] * g[1] - q3 * g[2])$
25: $qNew[1] \leftarrow q[1] + (-q[0] * g[0] - q[2] * g[1] - q3 * g[2])$
26: $qNew[2] \leftarrow q[2] + (-q[0] * g[0] - q[1] * g[1] - q3 * g[2])$
27: $qNew[3] \leftarrow q[3] + (-q[0] * g[0] - q[1] * g[1] - q[2] * g[2])$
28: $qNew \leftarrow normalize(qNew)$ {Normalise quaternion}
29: return $qNew$
```

Algorithm 2 accelerometerError

```
Require: $Input : a[], q[]$
Ensure: $e_v$
1: $v[]$ { define 3 dimensional array for the gravity field vector}
   {Compute the estimated direction of gravity as shown in 3.2.7 }
2: $v[0] \leftarrow q[1] * q[3] - q[0] * q[2]$
3: $v[1] \leftarrow q[0] * q[1] + q[2] * q[3]$
4: $v[2] \leftarrow q[0] * q[0] - 0.5 + q[3] * q[3]$
   {The error $e_v$ is the cross product between estimated direction and measured direction of field vectors}
5: $e_v[0] \leftarrow (a[1] * v[2] - a[2] * v[1])$
6: $e_v[1] \leftarrow (a[2] * v[0] - a[0] * v[2])$
7: $e_v[2] \leftarrow (a[0] * v[1] - a[1] * v[0])$
8: return $e_v$
```

Algorithm 3 magnetometerError

```
Require: $Input : m[], q[]$
Ensure: $e_b$
1: $h[]$ {define the magnetic field vector of the earth}
2: $b[]$ {define reference direction of the earth's magnetic field}
3: $b_s[]$ {define estimated direction of the magnetic field in the sensor frame}
   {Compute reference direction of Earth's magnetic field}
4: $h[0] \leftarrow 2 * (m[0] * (0.5 - q[2] * q[2] - q[3] * q[3]) + m[1] * (q[1] * q[2] - q[0] * q[3]) + m[2] * (q[1] * q[3] + q[0] * q[2]))$
5: $h[1] \leftarrow 2 * (m[0] * (q[1] * q[2] + q[0] * q[3]) + m[1] * (0.5 - q[1] * q[1] - q[3] * q[3]) + m[2] * (q[2] * q[3] - q[0] * q[1]))$
6: $b[0] \leftarrow sqrt(h[0] * h[0] + h[1] * h[1])$
7: $b[2] \leftarrow 2 * (m[0] * (q[1] * q[3] - q[0] * q[2]) + m[1] * (q[2] * q[3] + q[0] * q[1]) + m[2] * (0.5 - q[1] * q[1] - q[2] * q[2]))$
   {Compute estimated direction of magnetic field as shown in 3.2.14 }
8: $b_s[0] = b[0] * (0.5 - q[2] * q[2] - q[3] * q[3]) + b[2] * (q[1] * q[3] - q[0] * q[2])$
9: $b_s[1] = b[0] * (q[1] * q[2] - q[0] * q[3]) + b[2] * (q[0] * q[1] + q[2] * q[3])$
10: $b_s[2] = b[0] * (q[0] * q[2] + q[1] * q[3]) + b[2] * (0.5 - q[1] * q[1] - q[2] * q[2])$
   {Compute cross product between estimated direction and measured direction of field vectors}
11: $e_b[0] \leftarrow (m[1] * b_s[2] - m[2] * b_s[1])$
12: $e_b[1] \leftarrow (m[2] * b_s[0] - m[0] * b_s[2])$
13: $e_b[2] \leftarrow (m[0] * b_s[1] - m[1] * b_s[0])$
14: return $e_b$
```

3.3 Position Estimation

3.3.1 Gravity Compensation

In order to estimate the path of a golf club during a swing, we can use the accelerometer of the IMU that is attached to the club. The accelerometer will reflect motion of the IMU system by outputting acceleration at sample n in each of its axes represented by the last three components of quaternion ${}^S\hat{a}_n$. That acceleration has to be integrated twice to obtain displacement. This method would work correctly only if the device was positioned so that one of its axes would capture the full force of gravity $1g$ and that force of $1g$ is subtracted from that axis before integration is performed. However, the device will change orientation during a swing, meaning that the force of gravity $1g$ will be distributed in each of the three axes of the accelerometer. That means that the accelerometer's raw output is a combination of acceleration due to motion and gravitational force. Therefore, we need an acceleration vector called $\hat{d}_n$ that represents the dynamic acceleration of the IMU. Dynamic acceleration is only acceleration due to displacement. We know that we can obtain the expected gravity vector ${}^S\hat{v}_n$ at the n-th of the sensor fusion algorithm as demonstrated in equation 3.2.7. We can therefore subtract ${}^S\hat{v}_n$ from the raw acceleration readings ${}^S\hat{a}_n$, thus getting the expected dynamic acceleration in all three axes of the sensor:

$$\hat{d}_n = {}^S\hat{a} - {}^S\hat{v}_n \tag{3.3.1}$$

where we can ignore the first element of quaternion ${}^S\hat{a}$ and quaternion ${}^S\hat{v}_n$ as only their last three components contain gravitational force estimations.

3.3.2 Displacement Estimation

In order to approximate the position S of the IMU device over time, its speed has to be calculated. The velocity over time is calculated by integrating the dynamic acceleration values for each sample n:

$$v = \int_0^n d \tag{3.3.2}$$

The position of the device therefore would be the result of the integration of the velocity v:

$$S = \int_0^n v = \int_0^n \int_0^n d \tag{3.3.3}$$

This method means that acceleration readings must be integrated twice to get displacement. There is a sample time Δt between consecutive readings. The method described above is theoretically correct but would not work so well in practice because it requires very high sampling rate, in order to make Δt very small. Even when optimized for maximum speed the sample rate of the IMU will not exceed 400 Hz. Therefore, to better estimate the position of the IMU, rectangular midpoint integration will be performed: [18]

$$\hat{v} = \int_0^n \hat{d} \approx d_{n-1} + (\frac{d_n + d_{n-1}}{2}).\Delta t \tag{3.3.4}$$

$$\hat{S} = \int_0^n \hat{v} \approx v_{n-1} + (\frac{v_n + v_{n-1}}{2}).\Delta t \tag{3.3.5}$$

where n is the sample number and Δt is the time between two consecutive readings.

4
Golf terminology and GolfSwing Application

4.1 Golf Swing phases

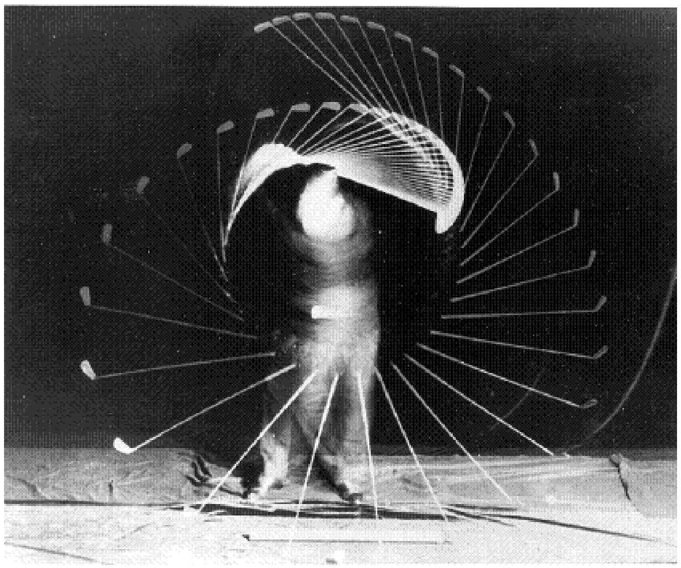


Figure 4.1.1: The path of a golf club during a typical full golf swing [24]

Figure 4.1.1 provides a picture of a golf club during a swing and figure 4.1.2 presents the composition of a golf club. Small differences in the orientation of the clubface during impact with a ball will affect the golf ball's path. In addition, the motion of the club immediately after impact with the golf ball has a major role in the accuracy of the golf shot. The whole process that a golf player goes through including body alignment and preparation is not of key importance for this paper. The motion of the golf club during a golf swing is the focus of this project as it can be evaluated using the novel hardware system. That motion can be categorized in four phases: back swing, down swing, contact and follow-through.

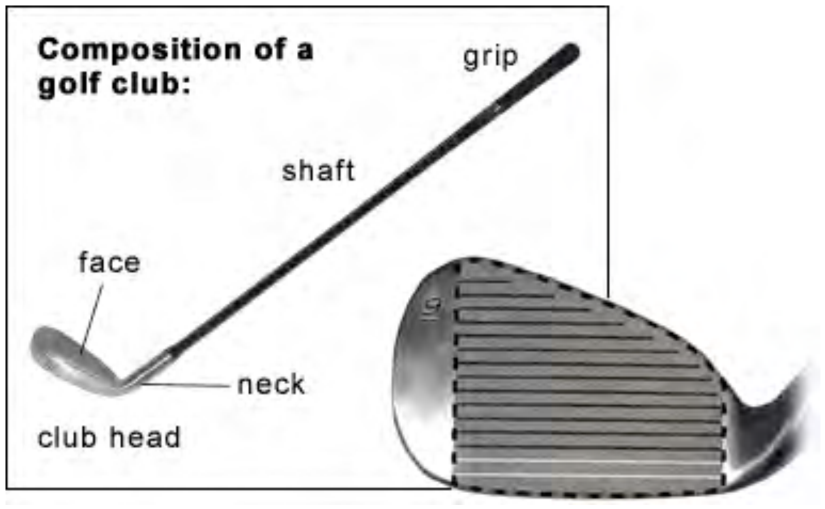


Figure 4.1.2: Composition of a golf club
[27]

The backswing begins as the player moves back the club that is initially placed perpendicular to the ground and next to the golf ball. That motion continues in even tempo until the club is parallel to the ground. The down swing is the next step and it should repeat the same movement of the back swing, but in opposite direction. The down swing of course is very fast and can happen in less then half a second. The moment of the impact of the club face with the ball is called the contact phase. It is a crucial moment, since that is when the momentum of the club is transferred to the ball. The orientation of the clubhead and the angle of attack are very important during the contact. The last phase is the follow-through. During that last phase, the direction of the path of the club can alter substantially the golf ball's final location. Note that on average the backswing takes around 0.8 seconds and the down swing takes 0.3 seconds. [9] That makes for a very short time frame of recording data using the prototype device. The next subsection elaborates on the follow through phase and its relevance to a golf player. The phases described above are used to decide on the way feedback is given to the golf player and the way sensor data is analyzed.

4.1.1 Follow-through phase

During the follow-through phase, right after the impact with the ball, the club can move in multiple distinct directions as depicted in figure 4.1.3. That movement can be put in two major categories: hook and slice. A hook occurs when the clubface follows an inside-out swingpath and thus flies severely from right to left for a right-handed player. A slice occurs when the clubface follows an outside-in swingpath and thus flies severely from left to right for a right-handed player. Skilled players can hook or slice the ball at will, but most commonly they are misplayed shots. [11]

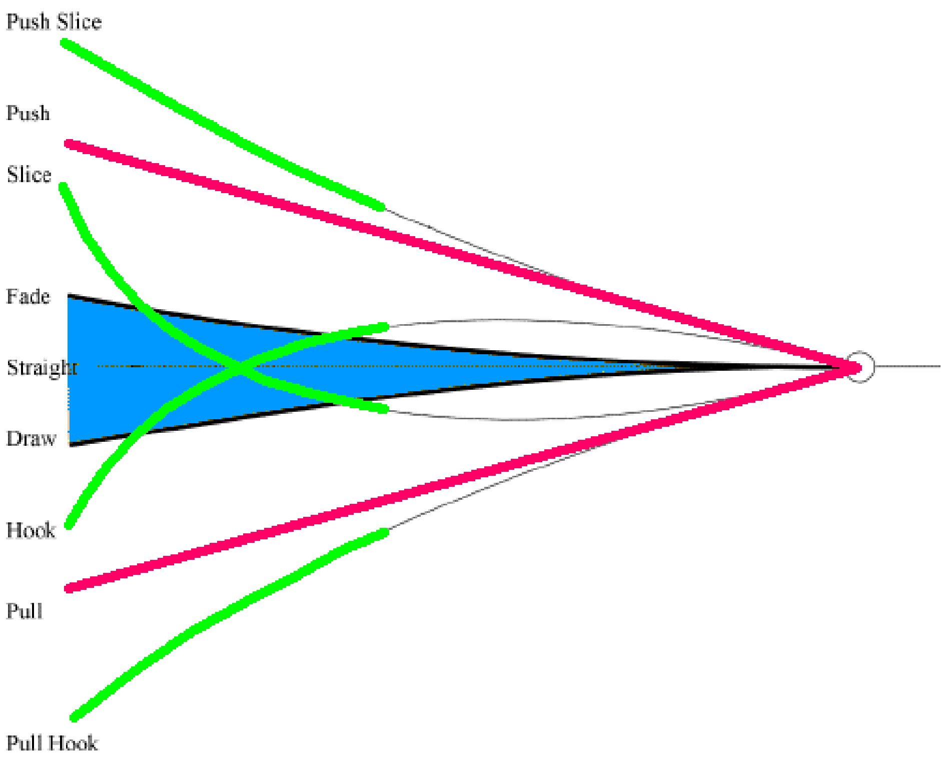


Figure 4.1.3: Possible paths of the clubhead during the follow-through phase of a golf swing

[19]

4.2 GolfSwing Application

4.2.1 Processing Application Overview

In this section, we introduces an application called GolfSwing, which interfaces with the IMU prototype and visualizes a replay of a golf swing. The application can also determine if a shot was hook or sliced using the follow-through phase data obtained from the IMU. This is a top-level overview of the application that demonstrates what functions it needs to perform to be useful for a golf player. I wrote the application using Processing. Processing is an open source programming language and environment for people who want to create images, animations, and interactions. [23] Processing is a good choice for this application because it supports OpenGL, which is used to graphically represent the motion and path of the golf club after a golf swing. Figure 4.2.1 shows the interface of the GolfSwing application used in this project. The user interface was designed with simplicity and functionality in mind. A user can choose between the following options:

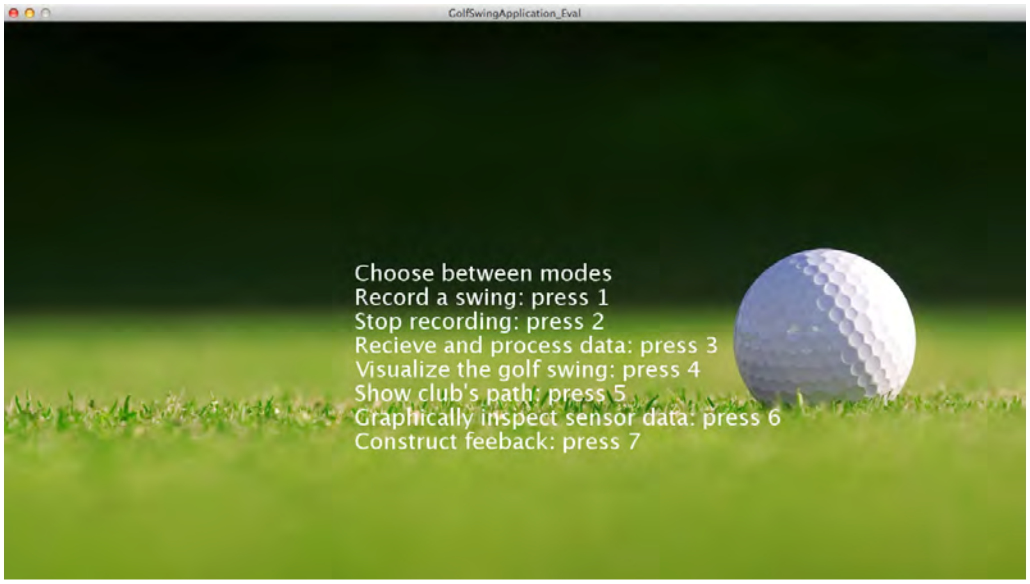

GolfSwingApplication_Eval
Choose between modes
Record a swing: press 1
Stop recording: press 2
Recieve and process data: press 3
Visualize the golf swing: press 4
Show club's path: press 5
Graphically inspect sensor data: press 6
Construct feeback: press 7

Figure 4.2.1: GolfSwing application and its menu.

1. Record a swing - The attached to a club IMU would start recording data. A second after its vibration sensor detects impact the IMU will stop recording and notify the application that it is ready to send the data over bluetooth.

2. Stop recording - Make the IMU stop recording

3. Receive and process data - Ask the IMU for the recording data, receive it and save it to a CSV file for reference. After that, process the data and save it to another CSV file containing the clubhead's position and orientation for each sample.

4. Visualize the golf swing - Visualize the motion of the golf swing with chosen replay speed using OpenGL API

5. Show club's path - Shows the whole path of the clubhead and the IMU by connecting the points representing their locations for each sample.

6. Graphically inspect sensor values for data analysis - Show the acceleration and velocity change over time on a graph for data analysis.

7. Construct feedback - determine if the shot hooked or sliced.

4.2.2 Feedback on a golf swing

In order to give constructive feedback, the golf swing application use the concepts of hook and slice. To determine the specific hook or slice, the change in orientation and displacement following the impact with the ball are checked. It is sufficient to trace the change in the x and y position of the clubhead for half a second after the impact with the ball. Thus the amount of movement of the club sideways results in a feedback score in the range from 1 to 10, where 1 would indicate a straight shot and 10 would indicate a maximum slice or hook movement. Figure 4.1.3 shows some of the possible movements of the club after impact. Using the latter technique, each of these movements could be approximated. In addition, the GolfSwing application can calculate the velocity of the club at the moment of impact and the maximum velocity, throughout the whole swing.

4.3 IMU System Data Recording Process Overview

Once the IMU microcontroller is notified to start recording it gets the raw readings from the accelerometer, gyroscope and magnetometer. Then it performs the sensor fusion algorithm on the readings. I adapted a library called FreeIMU that implements the sensor fusion algorithm for Arduino [31]. I modified it to work with 32-bit micro controllers including Teensy 3.0, by optimizing its drivers for the sensors. In addition to that I added gravity compensation to the accelerometer's raw readings to extract dynamic acceleration.

While recording, the microcontroller saves on the MicroSD Card the following: a quaternion array q, dynamic acceleration array d and sampling time float t for each sample. Once the IMU system stops recording it reads the data from the saved file on the MicroSD card and sends to a processing application over bluetooth. The average size of that data is 32 bites. The average sample rate of the configuration when data is logged is 400 Hz. If the data is not saved on the MicroSD card the sample rate is 900 Hz.

5
Sensor Performance Optimization and Data Processing Algorithm

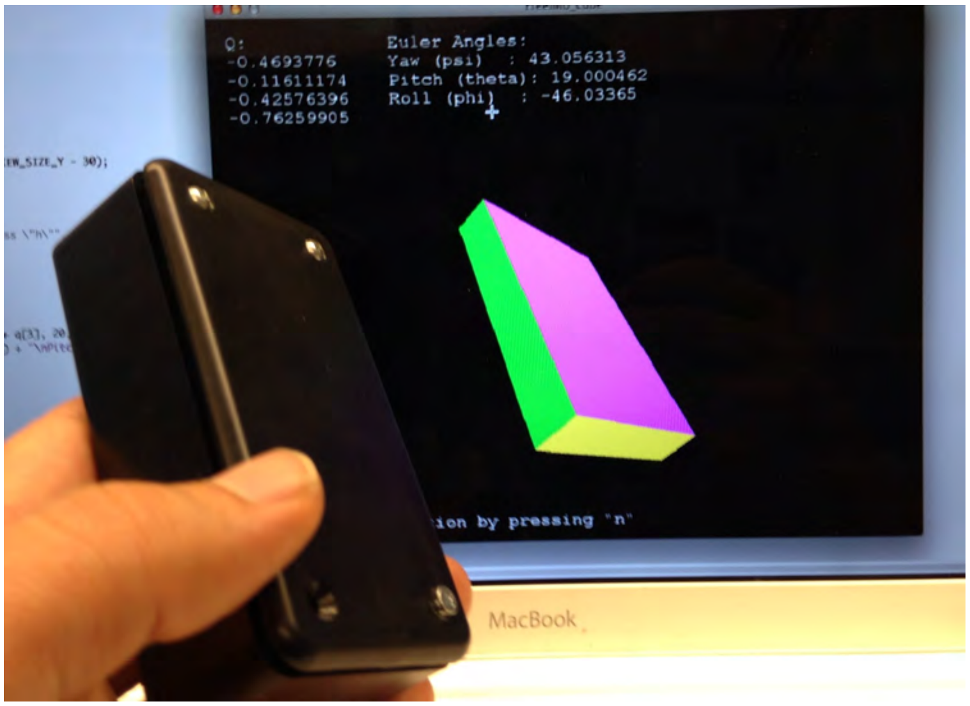
This chapter contains a discussion of the performance of the IMU system's individual sensors. The output of the MARG sensor fusion algorithm, discussed in section 3.2.2, is processed with several techniques to achieve optimum results in a golf swing trajectory prediction. We also investigate the prototype's most fit attachment location on a golf club. Each component of the IMU system is inspected and improved independently in order to construct an algorithm suitable for orientation and displacement estimation of a golf clubface during a swing. Note that we assume that in any case the device will not record data for more than 20 seconds. After all, the process of backswing and downswing takes less than 2 seconds on average. [9]

5.1 Orientation and displacement estimation without filters

Our final sensor fusion algorithm returns the IMU's quaternion orientation estimation - ${}^{S}_{E}\hat{q}_{e,n}$, dynamic acceleration vector $\hat{d}_n$ and time step Δt between sample n and $n-1$. It is important to investigate the accuracy of that data independently, so that any of its components that are prone to errors can be optimized using filtering.

5.1.1 Orientation estimation without filtering

To evaluate the orientation estimation capabilities of the IMU, I kept it stationary for 20 seconds and observed its Euler angles estimation over time.



```
Q:                Euler Angles:
-0.4693776        Yaw (psi)    : 43.056313
-0.11611174       Pitch (theta): 19.000462
-0.42576396       Roll (phi)   : -46.03365
-0.76259905
```

```
ion by pressing "n"
```

Figure 5.1.1: Determining the orientation of the device and displaying it using Processing

Figure 5.1.2 shows the drift of the estimated roll , pitch and yaw angles of the IMU for 20 seconds. As depicted in the graph, the drift accumulates to no more than 5 degrees. The IMU system's orientation is very stable mainly because it is obtained using the MARG sensor fusion algorithm shown in section 3.2.2 and because the device has 9 degrees of freedom. The addition of a magnetometer is essential, since with the magnetometer output, the IMU system can sense its orientation in reference to the earth's frame. If no magnetometer were present the IMU would not be able to correct its yaw estimation.

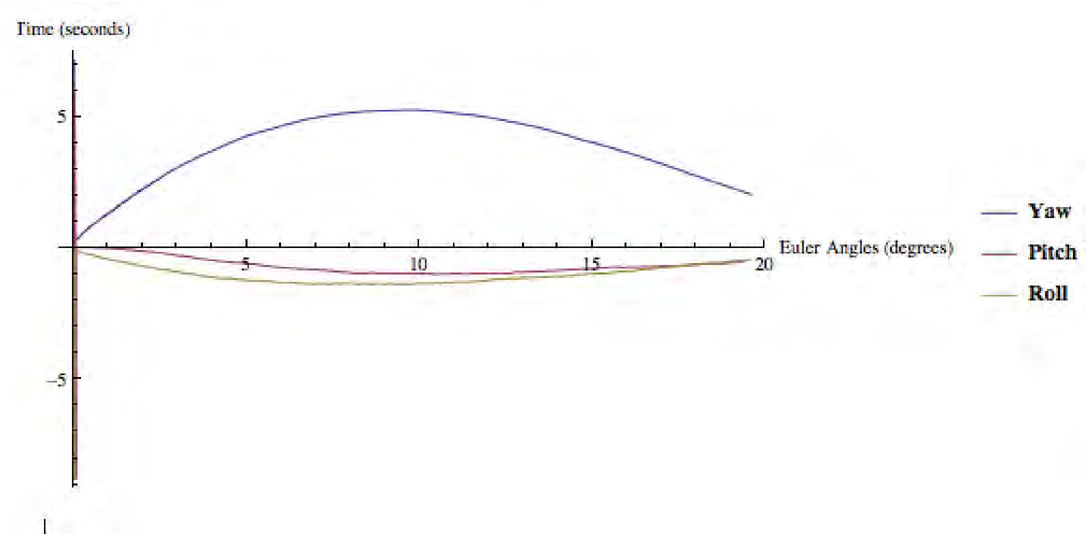


Figure 5.1.2: Yaw Pitch Roll estimations when IMU is placed stationary

So far the IMU system indicates successful orientation capabilities for our needs. However, in order to record a golf swing, the device that is attached to a club has to change orientation rapidly. Evaluation of its orientation sensing accuracy during rotational motion is demonstrated below. The IMU is placed on an actuator that can rotate its axle any chosen angle. (Figure 5.1.3) The point of the experiment is to compare the approximated orientation to the actual orientation of the device over the course of several axle rotations. The actuator axle repeats the following process over time: turn clockwise 180 degrees, instantly change direction and turn counter clock wise 180 degrees, delay for roughly a 0.8 seconds and repeat the process.

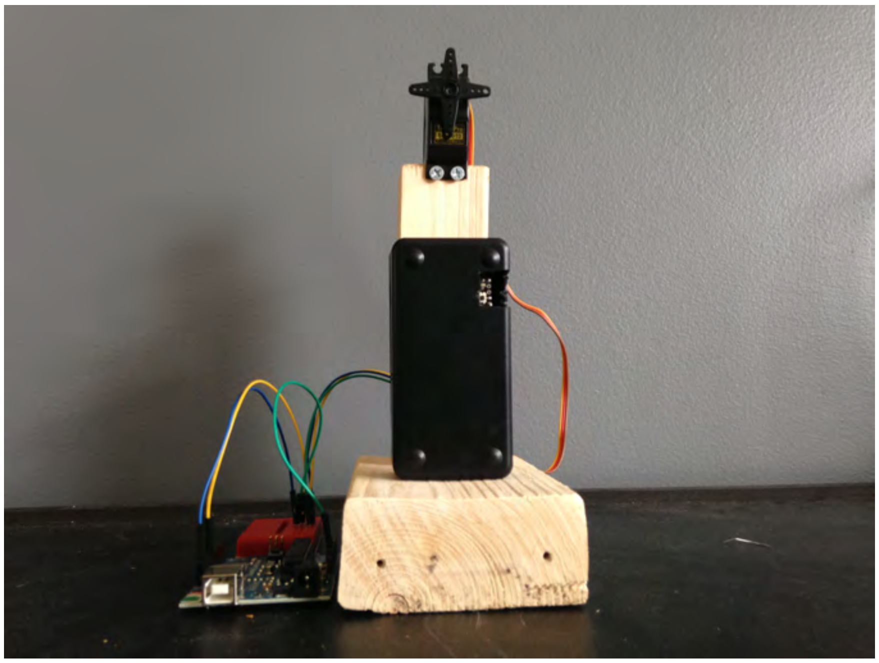


Figure 5.1.3: Actuator setup used to rotate the IMU

By rotating the IMU we can evaluate its orientation sensing accuracy

Figure 5.1.4 shows the result of that experiment. The figure reflects the axle rotation process and also demonstrates that the device has good orientation estimation capabilities in a dynamic setting. After 8 consecutive turns, the IMU still has accurate estimation of its initial heading and attitude at the beginning of the experiment.

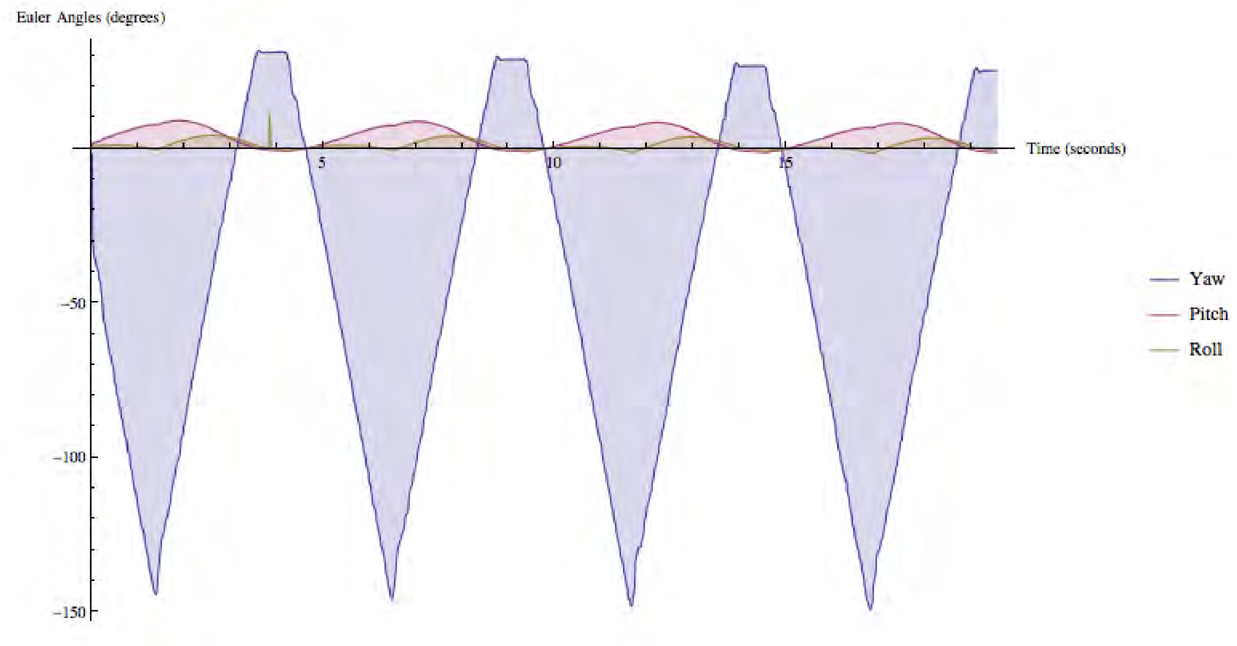


Figure 5.1.4: Euler angles estimation during actuator rotation

The successful orientation estimation of the IMU system over time implies that at the moment of impact with a golf ball the device can theoretically sense the club face's orientation with minimal error if it has been recording data for less than 20 seconds. We can safely assume that the orientation sensing process does not need filtering because the angular velocity is only integrated once and because in its current configuration the IMU is capable of very effective orientation estimation.

5.1.2 Position estimation without filtering

Ideally, the displacement of the IMU would be the result of integrating twice the components of the dynamic acceleration vector $\hat{d}$ as shown in section 3.3.2. However, there is a potential problem with employing only that technique. When double integrating constant acceleration, errors increases with the square of time because the error after 20 seconds is 400 times greater than that at 1 second. Any small offset errors in the acceleration measurement will quickly produce an intolerable error level. [3] In addition to that, all position estimation are based entirely on the dynamic acceleration output $\hat{d}$. The process of acquiring dynamic acceleration described in Section 3.3 can lead to additional errors because any difference in the IMU's orientation estimation and its real orientation will lead to inaccurate estimation of the gravity vector ${}^{S}\hat{v}_n$. Since the components of the gravity vector are removed from the accelerometer output vector ${}^{S}\hat{a}_n$, if the gravity vector estimation is not fully accurate, error will accumulate.

Finally, figure 5.1.5 depicts the expected position of the device based on double midpoint rectangular integration when the IMU prototype is placed stationary on a flat surface. Figure 5.1.5 shows that after only 6 seconds the expected displacement of the system in just its x axis is more than 30 meters.

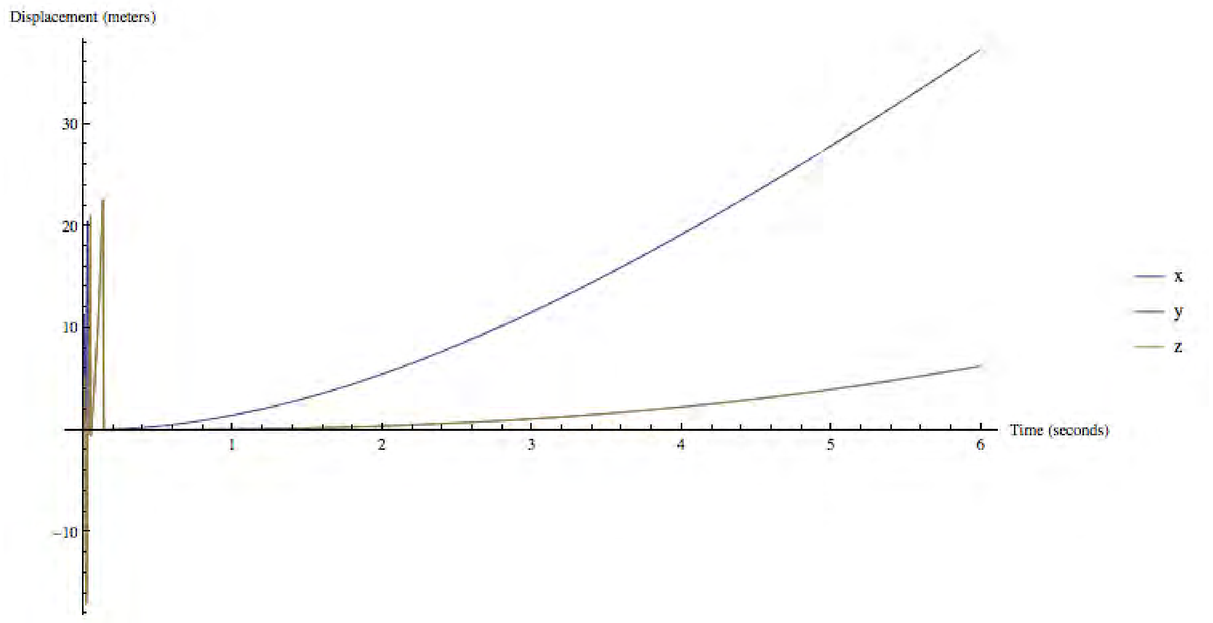


Figure 5.1.5: IMU's displacement over time without any filtering

It is clear that non filtered data would not be reliable over a time frame of more than a few seconds. Demonstrating the displacement estimation of the system when it undergoes motion is unnecessary because the error is increasing so rapidly even when it is not moving. A filtering technique must be applied to the process of double integrating dynamic acceleration. The next section proposes a methodology for that purpose.

5.2 Displacement estimation optimization

It is necessary to filter the dynamic acceleration output even though that can lead to less precise displacement calculations. Filtering reduces the risk of calculating a position estimation that is completely inaccurate compared to the actual position of the device. In other words, it is better to estimate displacement that contains the general motion of the IMU's physical movement versus displacement estimation that becomes useless after less than a second. Therefore, we propose a methodology for processing dynamic acceleration vector $\hat{d}$ for each sample from 0 to n.

1. Having input $\hat{d}_n$, we want to compute position vector $\hat{s}_n$ and velocity vector $\hat{v}_n$. For a given dynamic acceleration $\hat{d}_n$, we first check if the absolute value of any of the elements of the vector are below a threshold. That threshold is $0.1g$. If any of the components $\hat{d}_{xn}$, $\hat{d}_{y_n}$ or $\hat{d}_{zn}$ of the vector $\hat{d}_n$ have a gravitational force measurement whose absolute value is below the threshold, we set that component to 0. We do that because we want to filter erroneous acceleration output.

2. Integrate $\hat{d}_n$ to obtain velocity. We then check if the absolute value of the acceleration from the previous 10 samples of dynamic acceleration $\hat{d}$ are below the threshold 0.1g. If that is the case than we set the value in the axis of the velocity vector $\hat{v}$ to 0. We do this because we assume that the device is stationary and that it is not changing direction. We do not want to integrate velocity $\hat{v}_n$ when the device is not moving because position estimation would quickly produce intolerable error.

3. Finally we integrate the velocity to get position $\hat{s}_n$ at sample n.

Figure 5.2.1 shows the x, y and z displacement estimation with the filtering techniques described above when the IMU is placed stationary on a flat surface. Ideally over 6 seconds displacement in each axis should be 0. That is precisely the case as figure 5.2.1 demonstrates. Since we are resetting the acceleration whenever it bellow 0.1 g any error is filtered before any integration is performed. It is safe to have a threshold of 0.1 g as during golf swing the acceleration of the golf club is well above $5g$, meaning that we will not reset acceleration due to the actual club's displacement.

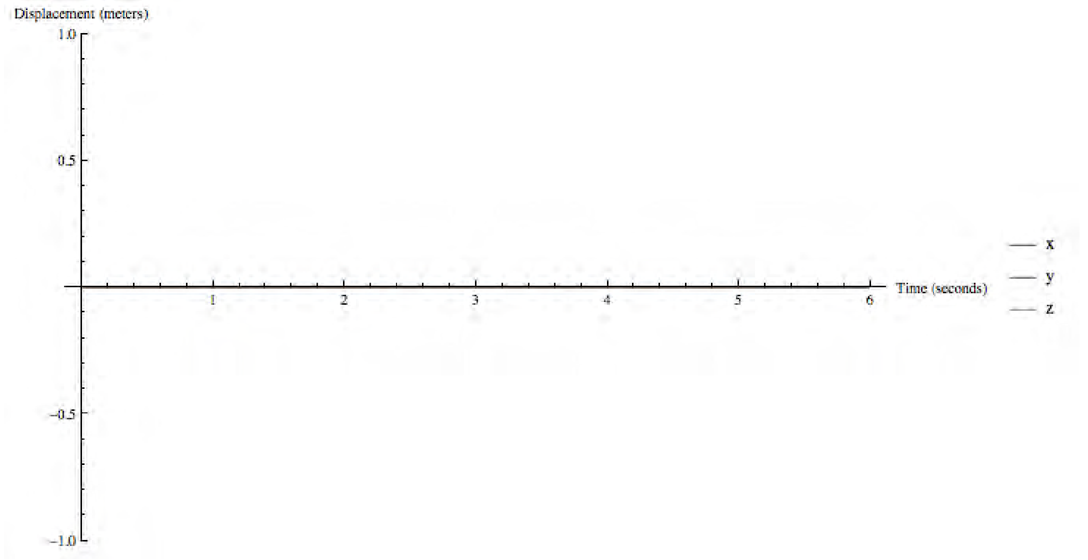


Figure 5.2.1: IMU's displacement over time with filtering

5.3 IMU Placement on a Golf Club

The IMU prototype has to be physically attached to a golf club to record data during the swing. The average length of a golf club is 40 inches [12] meaning that there are multiple locations where the device can be attached. In this section, two possible placement locations for the IMU system are examined : the golf club's grip area at the top and the club head's area at the bottom. The two different locations of the IMU would involve two different procedures for displacement estimation of the club head.

5.3.1 IMU at the head of the club

If the IMU system is placed at the bottom of the golf club, the clubhead's position estimation will depend entirely on the double integration of acceleration values. In addition to that, even a prototype of small size and weight would negatively impact the performance of the golf swing, as it adds extra weight to the clubhead. Golf clubs are tuned for a specific weight distribution and even a small addition of mass at the end of the club will alter the desired flight of the ball. For these reasons reasons, having the IMU at the bottom of golf club is not a desirable option.

5.3.2 IMU at the grip of a club

The IMU can easily be attached right below the grip of a golf club. The length of a golf club grip is roughly 0.25 meters therefore the IMU would be placed at 0.3 meters from the top of the club. Placed at that distance, it will not affect the club's weight distribution as much as it would at the the club head. Having the IMU near the grip means that the position estimation of the club head will be a combination of the double integration of acceleration data and a projection of the club shaft based on the IMU's orientation.

5.4 Final Displacement and Orientation Estimation Processing Algorithm

This section presents an overview of the process of estimating the position and orientation of a golf club head with the IMU system attached below the club's grip. Consider that the IMU is attached on a golf club below the club's grip. In that case the distance from the IMU's accelerometer sensor to the club head would be denoted by k. The length of the club itself is 1 meter. The cartesian vector denoted by $\hat{S}$ is the vector containing the final x, y and z coordinates of the clubhead in three-dimensional space.

Since the IMU system is placed below the grip, its orientation will be used to estimate the three-dimensional coordinates of the clubhead that is at distance k from the IMU's accelerometer. As described in 3.1.6, we can rotate a vector of certain length using a quaternion. Lets construct such a vector $\hat{p}_{in}$ of length k.

$$\hat{p}_{in} = (0, 0, 0, -k) \tag{5.4.1}$$

In this case having $-k$ as the last component of $\hat{p}_{in}$ means that the the projection of the clubhead would be an extension of the z axis of the device. The vector $\hat{p}_{out}$ is vector $\hat{p}_{in}$ of a length k, rotated using normalized quaternion $\hat{q}$. Quaternion $\hat{q*}$ is the conjugate of $\hat{q}$. The first element of the two 4 dimensional vectors $\hat{p}_{out}$ and $\hat{p}_{in}$ is 0.

$$\hat{p}_{out} = \hat{q} \otimes \hat{p}_{in} \otimes \hat{q*} \tag{5.4.2}$$

The vector $\hat{p}_{out}$ will contains the coordinates of the end of the club if it rotates around a fixed point and if the accelerometer of the IMU system is placed on that fixed point. Since the IMU must be placed on the grip the final location of the clubhead $\hat{S}$ is the sum of the vector $\hat{p}_{out}$ and the vector containing the coordinates of the IMU's accelerometer.

The final heading and displacement algorithm returns the clubface's estimated orientation ${}^{S}_{E}\hat{q}_{e,t}$ and position $\hat{S}$. We simply sum the vector containing the Cartesian coordinates of the IMU's accelerometer $\hat{s}_{imu}$ with the vector containing the coordinates of the club head's expected position $\hat{p}_{out}$ to obtain the final position vector $\hat{S}$:

$$\hat{S}_n = \hat{s}_{imu} + \hat{p}_{out} \tag{5.4.3}$$

The vector $\hat{S}_n$ represents the final location of the clubhead in 3D space for each sample n. Its coordinates are saved in a designated data structure using the GolfSwing Processing application and thus the golf swing can be visualized.

6
IMU Prototype Evaluation with Simulations

6.1 Process of Evaluation

In order to evaluate the IMUs performance, an environment with proper golf swing simulation equipment for conducting tests is essential. The output of the final displacement algorithm in section 5.4 has to be empirically compared to its expected true result. We need to be able to put the device through approximately the same type of motion and orientation change, multiple times. In addition to that, the position of the IMU at any given time must to be known. The system's performance and other technical and economical aspects are evaluated in this chapter.

6.2 Golf Swing simulation using Pendulum

The IMU's golf swing recording and analyzing capabilities have to be considered and evaluated. A pendulum is an excellent candidate for golf swing simulations and empirical evaluation. A pendulum is a weight suspended from a pivot so that it can swing freely. When a pendulum is displaced sideways from its resting equilibrium position, it is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position. When released, the restoring force combined with the pendulum's mass causes it to oscillate about the equilibrium position, swinging back and forth. [22]

6.2.1 Nonlinear Driven Damped Pendulum

Pendulums are most often modeled with a a string and a mass at its end, making the force of friction and air resistance negligible. A string cannot accommodate the needs of this experiment because the IMU must be attached on a rigid rod. Using a rigid rod means that friction cannot be neglected. That means that the pendulum is underdamped. [10] A model for such a damped pendulum is presented here. The angular frequency of the damped pendulum is Ω. The frequency can be calculated using the period T of a pendulum:

$$\Omega = \frac{1}{T} \tag{6.2.1}$$

where T is in seconds and Ω is in $\sec^{-1}$. Equation 6.2.2 shows how to compute the expected angle θ of the rod over time. [10]

$$\theta = \theta_0 e^{-t/\gamma} \cos(\Omega t) \tag{6.2.2}$$

where the initial angle θ_0 and the expected angle θ are in radians and γ is the expected decay time. The constant γ depends on friction of the rod and air resistance but for our needs we can assume it is 0.5. [10] Finally we can compute the expected x and y position of a pendulum rod of length L given the rods angle θ:

$$x = L\sin(\theta) \tag{6.2.3}$$

$$y = L\cos(\theta) \tag{6.2.4}$$

A pendulum rod follows a circular trajectory. Nevertheless a pendulum is sufficient to effectively test the performance of the prototype IMU as the goal is to compare the output of the displacement and position estimation algorithm to some true data.

6.3 Simulation Pendulum Environment

I constructed a pendulum equipment that can be used to perform the same experiments multiple times. The pendulum's rod is made of wood and its length is 1 meter. The simulation involves releasing the pendulum rod at an angle θ of 90 degrees. That is a suitable release angle because we will simulate the down swing, the contact and the follow through phase of a golf swing. To ensure that during testing the IMU will be released at an angle of precisely 90 degrees I placed a metal piece positioned so that I can release the rod at exactly 90 degrees. The IMU system is attached so that the accelerometer sensor is at a distance of 30 cm from the rotation axle. That means that the IMU is below the grip of the club. Figure 6.3.1 shows the pendulum with the IMU attached at an angle of 90 degrees. Using a stopwatch, I timed the pendulum's period 20 consecutive times after I released it at an angle of 90 degrees. The average period of the pendulum was 1.86 seconds and that is the period based on which the pendulum rod's expected angle over time was computed using equation 6.2.2.

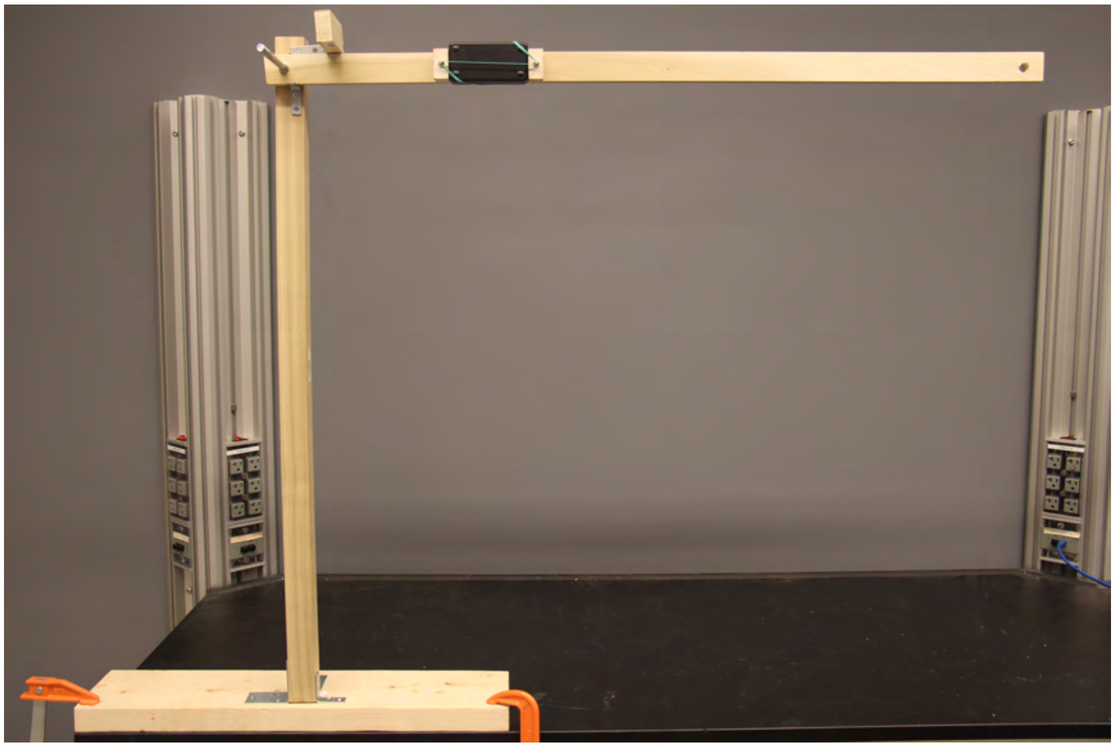


Figure 6.3.1: Pendulum equipment designed to simulate a simple golf swing

6.4 Experiment's Results

The golf swing simulation using the pendulum equipment was performed 40 times to observe the consistency of the accuracy of the IMU device. For each simulation the accuracy in each axis, and the path and position over time of the club in three-dimensional space was recorded. Data analysis was performed using a modified version of the GolfSwing application. A simulated pendulum object was created using the application. To measure the accuracy of a particular simulations we are comparing the coordinates of the vectors representing the end of the objects rod's position over time and the projected club head's position over time. Figure 6.4.1 through figure 6.4.4 depict histograms of the average accuracy in each axis for the 40 golf swing simulations that were conducted in the methodology described in section 6.3

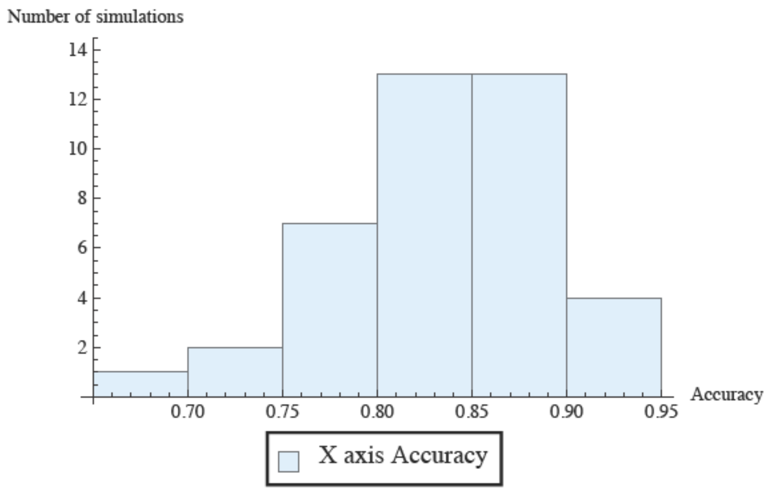


Figure 6.4.1: Histogram of the X axis accuracy for 40 golf swing simulations.

The average accuracy in the X axis is 82%.

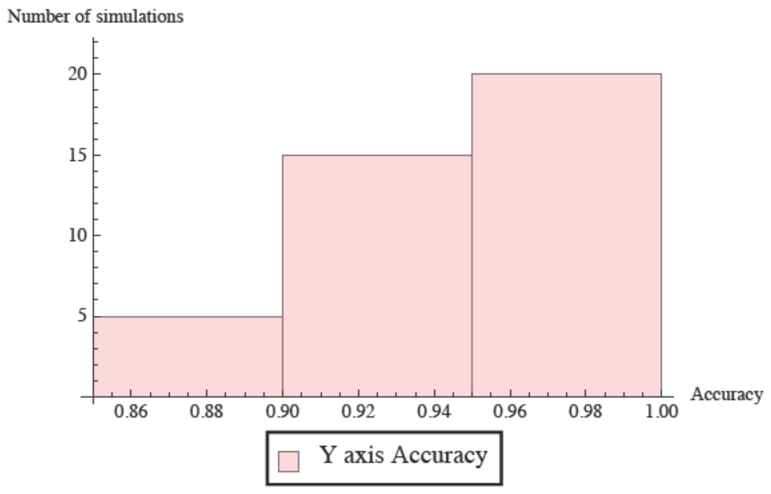


Figure 6.4.2: Histogram of the Y axis accuracy for 40 golf swing simulations

The average accuracy in the Y axis is 94%.

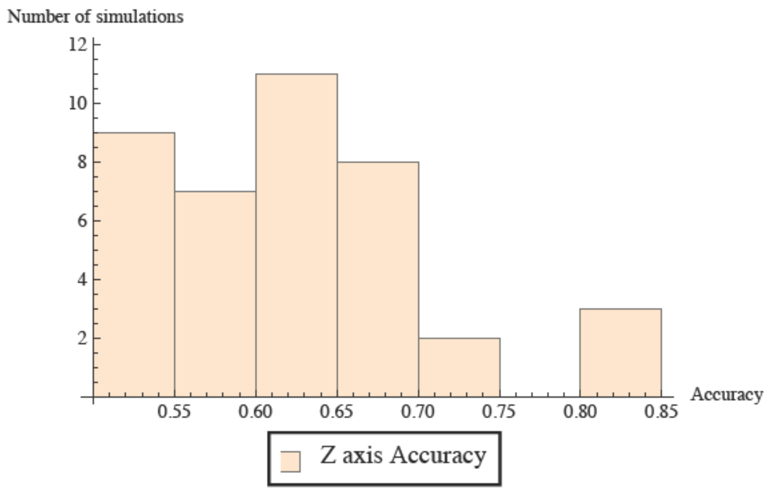


Figure 6.4.3: Histogram of the Z axis accuracy for 40 golf swing simulations

The average accuracy in the Z axis is 62%.

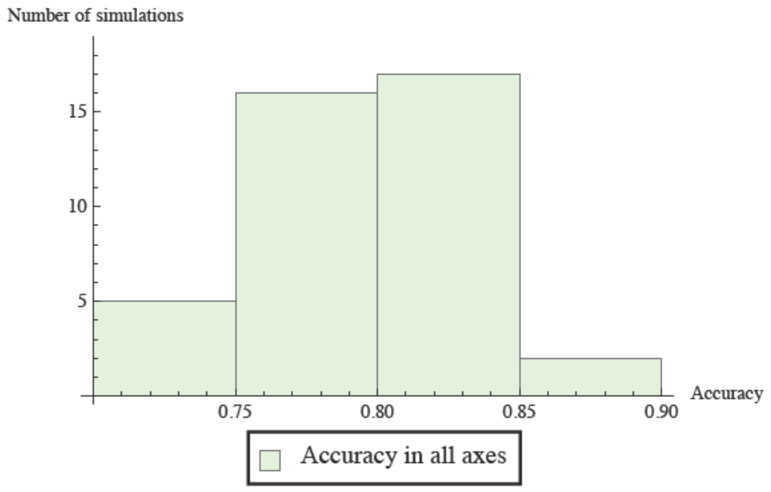


Figure 6.4.4: Histogram of the accuracy of all axes for 40 golf swing simulations

The average accuracy in all axes is 80%.

Figure 6.4.5 shows the average accuracy of all simulations for a half a period. Note that an accuracy of 1 means that two compared coordinates of the club head and the pendulum's rod are the same.

Figure 6.4.5: The accuracy for 40 golf swing simulations for 0.93 seconds.

The mean accuracy of all simulations is 80%. To inspect the performance of the club head position estimation algorithm in more depth we are taking a look at a swing simulation with average accuracy of 79%. Figure 6.4.6 shows the path of the pendulum versus the path of the club over the time of 0.93 seconds as that is half the period of the pendulum. For half a period, the physical pendulum simulates a down-swing and follow-through motion as in a real golf swing.

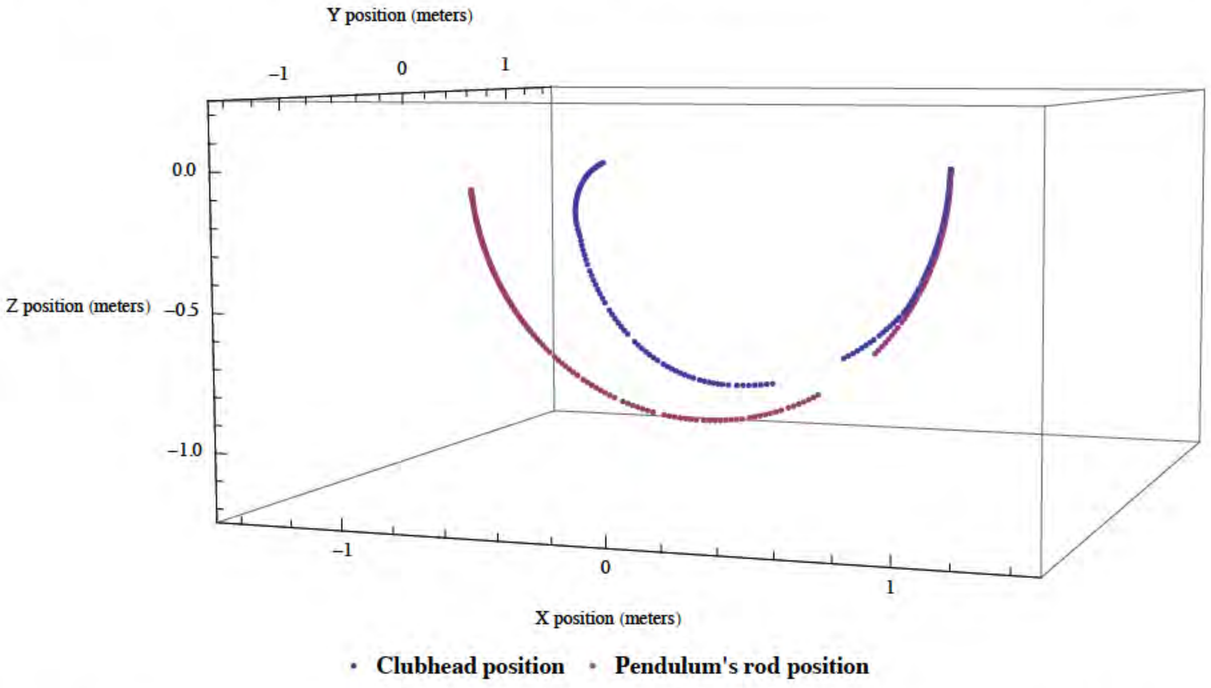


Figure 6.4.6: The path of a pendulum rod the club head path estimated using the pendulum experiment

The pendulum's rod is released at an angle of 90 degrees

Figure 6.4.7 shows the calculated x, y and z coordinates of the club head and the x, y and z coordinates of the simulated pendulum from the sampled experiment with accuracy of 79%

Figure 6.4.7: The path of the clubhead and the end of the pendulum rod of a simulation

Finally it is worth examining the IMU's displacements estimation capabilities over a period of more then a second. Figure 6.4.8 depicts the three-dimensional path of the clubhead and the pendulums path over 5 seconds.

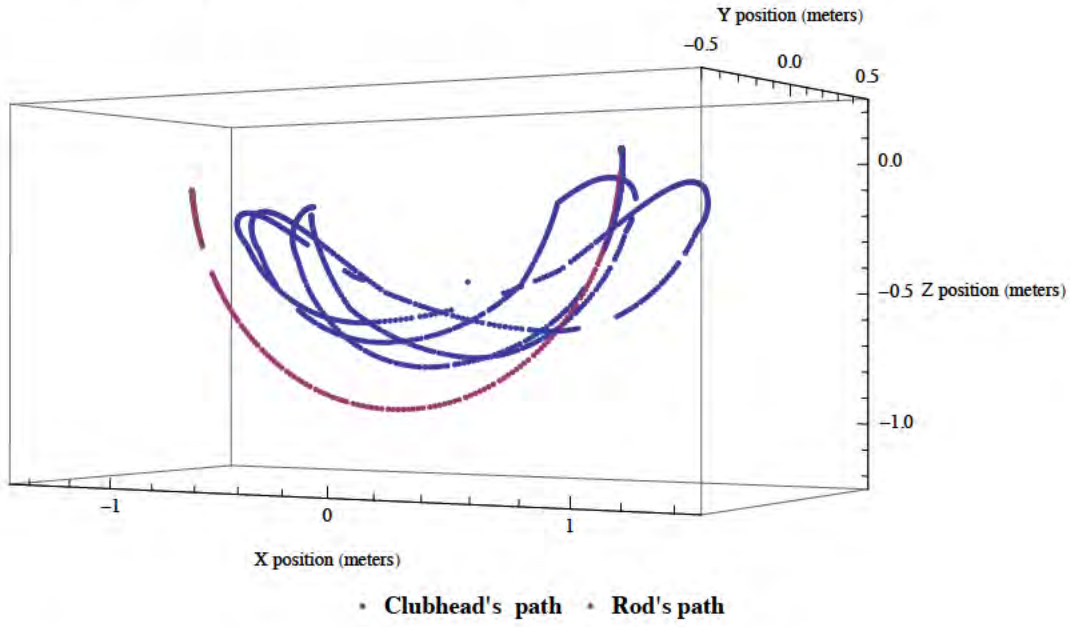


Figure 6.4.8: The expected path of rod end over 5 seconds on the pendulum

6.5 Discussion

6.5.1 Position estimation

The final displacement algorithm that computes the clubhead's position based on dynamic acceleration double integration and club shaft projection has an average accuracy in all axes of 80%. The accuracy in the x, y and z axes is 82%, 94% and 62% respectively. That is a good achievement, considering the multiple error factors affecting the process of data extraction: computing dynamic acceleration , MEMS sensors errors, and limited sample rate. During the simulation experiment, the IMU physically moves most in the z axis of the pendulum's frame. The z axis' accuracy is the lowest with an average accuracy of 62% percent. That could be attributed to the fact that, as the device is physically moving, its sample rate is not sufficient to acquire enough readings, for the double integration of dynamic acceleration. We apply filtering to the acceleration values but that does not contribute to reducing the magnitude of displacement when the IMU undergoes fast motion and experiences gravitational force that is higher than $0.1g$.

The IMU's clubhead projection based on the device's orientation seems to play a major role in representing the club head's path as shown in figure 6.4.6. It is important to note that as figure 6.4.8 shows, the path estimation of the clubhead for 5 seconds is fairly consistent. Although the pendulum's rod swings back and forward multiple times and the acceleration experienced by the accelerometer's axes changes drastically the estimated path of the clubhead is fairly accurate. That implies that in its current configuration the device could be used as an inertial navigation system, but some more advanced filtering methods should be applied to the accelerometers data to compensate for the sample rate of 400 Hz. Kalman filter is one such technique.

6.5.2 Other evaluation aspects

In this section, we include other factors involved in determining how well the IMU performs and its de facto practicality. Most of them come from the goal of the project outlined in Chapter 1. Its total cost is \$197 and it is constructed entirely using commercially available products. That means that it can be assembled and used for inertial navigation purposes or other applications. On the technical side, the number of degrees of freedom, size , weights and robustness are other important evaluation factors. The IMU has 9 DOF and is protected by a removable enclosure. It has Bluetooth capabilities which are nowadays standard for any computer or mobile device. That makes facilitating communication easy. Although the IMU is a prototype it is very small and light. Powered by battery it can work for more than 2 hours.

A company called Xsens produces motion and orientation tracking sensors used in several industries. Figure 6.5.1 provides a comparison between the technical characteristics of our prototype IMU and the company's entry level model MTi 10-series. [33]

	IMU prototype	Xsense IMU
Price	$197	$ $1,290
Degrees of freedom	9 DOF	6 DOF
Feature	Vibration Sensor, SD card adapter	
Communication	Bluetooth, USB	RS422 Interface
Accelerometer	16g	5g
Gyroscope	2000 deg/sec	450 deg/sec
Size	101x50x25 mm	57x42x23 mm
Sample Rate	400 Hz	400 Hz

Table 6.5.1: A comparison between the prototype IMU and the Xsense entry level model IMU

Although the device is just a prototype and does not come with the sophisticated software support offered from an established company, it is nevertheless significantly cheaper and easier to modify. It even has better technical specifications the the Xsense IMU. Overall the IMU prototype is a robust configuration, that can deliver performance especially if it's sensors output is processed according the specific use of the device.

7
Conclusion and Future Work

The goal of this project was to design and evaluate a hardware system that can be attached to a golf club and used to analyze a golf swing and its key parameters. A software that can interface the hardware system was created to visualize a golf swing trajectory and potentially give feedback to a golf player. The prototype system is easy to built and significantly cheaper compared to other products with similar characteristics. Its price of only $197 and the modularity of its components make it a great hardware for research involving inertial navigation and orientation sensing. We demonstrated that the prototype has excellent accuracy for attitude and heading estimation. The trajectory of a simulated golf swing was calculated with an average accuracy of 80%. Such accuracy is sufficient to determine if a golf swing is hooked or sliced, which is something that a golfer would like to know without the use of high speed cameras or a golf instructor.

The development and improvement of the IMU device can be continued as follows. The system can be evaluated using a more sophisticated equipment that can simulate a variety of golf swings including hooked or sliced shots. A faster data logging methodology for the IMU's output samples can be considered as that will significantly increase the sample rate of the configuration. Finally a more complex filtering method could be applied to the dynamic acceleration used to estimate displacement.

MEMS sensors would only become smaller and more efficient in the future as tethnology advances. Devices like the IMU proposed in this paper can become popular in many aspects of life. The system can be potentially used in other sports such as baseball or tennis. It can also be used in other industries since it can log large amounts of data and observe orientation change and displacement patterns.

Bibliography

[1] Analog Devices, *ADXL345: 3-Axis Digital Accelerometer*, `http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf`.

[2] Arduino, *Arduino Electronics Platform*, available at `http://www.arduino.cc/`.

[3] Andrew Blake and Graham Winstanley, *Deriving Displacement from a 3 axis Accelerometer*, available at `http://www.cem.brighton.ac.uk/research/cig/papers/Displacement\%20from\%20Accelerometer.pdf`.

[4] Eric Cope, *Estimating Human Movement Using a Three Axis Accelerometer* (2009), available at `http://cope-et-al.com/wp-content/uploads/2009/03/ericcopeqe2007.pdf`.

[5] James Diebel, *Representing attitude: Euler angles, unit quaternions, and rotation vectors*, Matrix (2006).

[6] A Lakshmi and Das Durga KC and Mohan, *Development Of DSP Based Magnetometer.*

[7] David F. Guillou, *Packaging MEMS*, `http://archives.sensorsmag.com/articles/1203/20/main.shtml`.

[8] Julian W. and Varadan Gardner Vijay K., *Microsensors, Mems and Smart Devices* (2001).

[9] Robert D and Cholewicki Grober Jacek, *Towards a biomechanical understanding of tempo in the golf swing*, arXiv preprint physics/0611291 (2006).

[10] Dr. Richard J. Gonsalves, *Nonlinear Driven Damped Pendulum*, `http://www.physics.buffalo.edu/phy410-505-2009/topic4/lec-4-4.pdf`.

[11] Golf-Information.info, *Golf Shots Terminology*, `http://www.golf-information.info/different-golf-shots-terminology.html`.

[12] Erica Henritz, *Standard length of golf clubs for women and men*, `http://www.livestrong.com/article/348393-standard-length-of-golf-clubs-for-women-and-men/`.

[13] Honeywell, *3-Axis Digital Compass IC HMC5883L*, 2013, `http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf`.

[14] InvenSense, *ITG-3200 Product Specification Revision 1.7*, 2011, `http://www.invensense.com/mems/gyro/documents/PS-ITG-3200A.pdf`.

[15] Rui Daniel Meng, *Design and implementation of sensor fusion for the towed synthetic aperture sonar* (2007).

[16] Douglas N. Arnold, *Mathematics That Swings: The Math Behind Golf*, `http://www.ima.umn.edu/~arnold/golf/golf-talk.html`.

[17] Jack B Kuipers, *Quaternions and rotation sequences*, Geometry, Integrability and Quantization **1** (2000), 127–143.

[18] Doron Levy, *Numerical Integration*, `http://www2.math.umd.edu/~dlevy/classes/amsc466/lecture-notes/integration-chap.pdf`.

[19] Marken-Golf.de, *Slice, Hook, Pull, Push, Fade, Draw, Straight*, `http://www.marken-golf.de/golftechnik/?id=140`.

[20] Sebastian OH Madgwick, *An efficient orientation filter for inertial and inertial/magnetic sensor arrays*, Report x-io and University of Bristol (UK) (2010).

[21] National Instruments, *Accelerometer Principles*, `http://zone.ni.com/devzone/cda/ph/p/id/12`.

[22] Princeton.edu, *Pendulum*, `http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Pendulum.html`.

[23] Processing, *Introduction to Processing*, `http://processing.org/`.

[24] Real-world-physics-problems.com, *The Physics Of A Golf Swing*, `http://www.real-world-physics-problems.com/physics-of-a-golf-swing.html`.

[25] SD FAT Arduino Library, *A FAT16/FAT32 Arduino library for SD/SDHC cards*, available at `https://code.google.com/p/sdfatlib/`.

[26] SparkFun Electronics, *9 Degrees of Freedom - Sensor Stick*, `https://www.sparkfun.com/products/10724`.

[27] SpinPro, *Golf Clubs*, `http://www.spinpro.de/uk/linspray1.htm`.

[28] Paul Stoffregen, *Teensy USB Development Board*, available at `http://www.pjrc.com/teensy/`.

[29] VectorNav Tehnologies, *Magnetometer*, `http://www.vectornav.com/support/library?id=83`.

[30] Fabio Varesano, *Using arduino for tangible human computer interaction* (2011).

[31] ———, *FreeIMU: an Open Hardware Framework for Orientation and Motion Sensing*, `http://www.varesano.net/projects/hardware/FreeIMU`.

[32] Oliver J Woodman, *An introduction to inertial navigation*, University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696 (2007), available at `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.html`.

[33] Xsens North America Inc., *MTi 10-series*, `http://www.xsens.com/en/general/mti-10-series`.