A Sensor-Augmented Golf Club

by

Yousaf Saeed, BSc (Hons)

A Dissertation submitted to the University of Dublin,

in partial fulfillment of the requirements for the degree of

M.Sc. Computer Science

2006

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Yousaf Saeed

September 15, 2006

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Yousaf Saeed

September 15, 2006

Acknowledgments

Thanks to my supervisor Dr. Mads Haahr, for his guidance and support throughout the duration of my time in Trinity. Thanks also to the rest of my lecturers from DSG, who have made a lasting impression on me. Thanks to my fellow classmates, for making it a great year.

Lastly, thanks to my parents Anwar and Zarina, for their continuing love and support.

Yousaf Saeed

University of Dublin, Trinity College

September 2006

A Sensor-Augmented Golf Club

Yousaf Saeed,
University of Dublin, Trinity College, 2006

Supervisor: Mads Haahr

Ubiquitous Computing paints the picture of a world, in which the purpose of a computer is to help you do something, whilst remaining relatively silent in the background. This paradigm of Ubiquitous computing has given birth to a substantial range of smart applications, which aim to make our experiences more calming and satisfying.

New sensory technologies and gesture recognition facilitates the deployment of applications for the sick and disabled, to provide them with a better quality of life. The same technologies allows us to interpret real world data and manipulate it to provide more adequate services, required for sports and training through the use of smart devices.

This Thesis presents "A Sensor-Augmented Golf Club", which is a smart golf club

created to interpret the gesture of a golf swing. Upon recognition, its associated application records and analyses the accumulated sensor data in real-time and provides the user adequate feedback in order to achieve a perfect swing.

Contents

List of Figures

Listings

Chapter 1

Introduction

"Ubiquitous Computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives." - Mark Weiser

1.1 Background

The field of Ubiquitous Computing inspired by Mark Weiser[17], is still youthful in age. For sure, he envisioned wireless sensor networks and miniaturised interconnected embedded appliances, but he also put forward the idea of sensor-augmented artefacts. These artefacts would be capable of sensing real world data, thus becoming more intuitive and context-aware, in terms of their interactions and their deployment of services for humans.

The new age of miniature sensors, has given birth to wireless sensor network applications, which make the environment more context aware in order to provide services. Throughout the last year, I have discovered new projects deploying smart spaces in environments such as office and homes.

Aside from the smart spaces, miniature sensors are also being used for the deployment of applications to aid sports and game training. This has inspired the work in this thesis.

1.2 Golf

In order to develop an application for a sport, one must have an underlying knowledge of its history and its structure.

Originally, it is said that Golf has been played for five centuries and it was thought to have originated from Scotland. The Dutch have also laid claims for its origins, believing that it was played first in 1297, in a city called "Loenen aan de Vecht". However, in January 2006 new evidence was uncovered to fuel the debate of its origins. Professor Ling Hongling of Lanzhou University unearthed records and drawings from the "Song" dynasty describing "Chuiwan", known to us as Golf. The records depicted equipment equivalent to woods, which are currently used in Golf except these clubs were inlaid with jade and gold, which again puts forward the idea that Golf itself was originally class orientated and geared towards the wealthy. In the meantime, a statement has been released from St. Andrews Golf organisation in Scotland claiming that modern day Golf as we know it, originates from Scotland. Contrary to the statement, we all

know that every sport has undergone an evolutionary process, in which the rules have been slightly altered and the equipment used has been upgraded to the state of the art, in order to further the sport. Take football for an example and ponder for a minute.

Golf indeed is classed as a sport, due to the strenuous routines and physical training of the body, which is required in order to naturally execute a swing with precision. It can be played individually, or indeed in teams, and is one of the few sports which is not executed on a fixed area of play.

Simply put, the aim is to navigate from the first hole to the eighteenth hole, using equipment known as clubs. The rules permit the player to carry fourteen clubs in total, consisting of irons, woods and a putter. Each player keeps track of his/her score, right through until the last hole and as a result, the player with the least shots wins.

1.3 The One Wood

The "Sensor-Augmented Golf Club" utilises a one wood driver. This club is used for teeing off, in order to gain a positional advantage on the fairway. It commands the longest shot in Golf and utilising this club can project the ball to a distance of up to 300 yards. The one wood is widely regarded as the hardest club to use and its execution requires a wider positional stance and a rotary movement of the shoulder, through the backswing and follow through of the shot.

1.4 Motivation

Golf requires practice for perfection and even pro golfers undergo routine tutoring through lessons. Deploying the correct technique is the holy grail for amateur golfers. Currently, the market is flooded with software utilities and simulators that claim to be like personal tutors and many prefer to use these, in the comfort of their homes. These simulators are absurdly expensive, costing anything from €25,000 - €50,000.

Given that sensor embedded artefacts have the power to capture and fuse real world data for analysis, I intended to seize this opportunity and develop the "Sensor-Augmented Golf Club", which would help the user to work on his/her golf swing via recognition and feedback. Realising the difficulties with the one wood, I thought there would be a place for such a club amongst enthusiastic amateur golfers.

1.5 Objectives

My dissertation objectives are:

- To record and store all of the orientation and calibrated dataset produced by a swing.
- To analyse the dataset with respect to an algorithm, which determines the level of accuracy achieved, in line with the dataset of a perfect swing.
- To recognise and analyse the backswing, teeshot and follow-through of a swing.
- To produce a report for the user indicating where his/her technique is failing and

to provide feedback on how to improve the technique based on errors made.

- To Analyse some of the upper body movement using an additional sensor during the swing, to aid the development of a correct technique.

1.6 Dissertation Outline

The structure of this Dissertation is as follows:

Chapter 2 State of the Art Review, is an overview of recent work which utilise the power of miniature inertial sensors, for applications related to sports training.

Chapter 3 Design Overview, gives an overview of the design and structure of the proposed system.

Chapter 4 Swing Analysis, this chapter describes specific parts of a golf swing, and the data necessary for analysis.

Chapter 5 Implementation, this chapter illustrates the technical aspects of the application.

Chapter 6 Evaluation, this chapter will evaluate the success of the application and provide some user scenarios.

Chapter 7 Conclusions, this chapter summarises the work presented and includes future work.

Chapter 2

State of the Art Review

2.1 Introduction

This section is intended to describe some of the latest work, which has been applied by sensor related applications for the Ubiquitous Computing paradigm. I will be presenting work which I believe has furthered the Ubiquitous application era, by facilitating some service, in order to help us achieve something with relative ease. I will also be focusing on work related to sports and fitness training. The intention here is to provide a clear understanding that sensory technologies allow us to have a very broad range of applications, which take in real world data for interpretation whilst providing a clever sustainable service, in order to be more intuitive than the standard applications prevalent in the past few decades.

The success of Ubiquitous computing and Pervasive applications[14], will be judged not only by their usefulness, but by their *transparency*, or simply put, their ability to

find a place to blend into our societies in a natural way, by being unobtrusive whilst facilitating a new means of interaction.

2.2 Application of Accelerometers in Sports Training

"New products and markets for the use of accelerometers in sports training have not yet been created. If one could obtain a greater understanding of sensor-based human performance measurement, such as the determination of a characteristic signature of the "perfect" movement, possible new products and markets may be created."

Above is the problem statement proposed by this particular research group. The intention was to measure the human performance of repetitive movements for rowing. This project is amongst the forefront of research work done with sensors and sports training. The research group from Analog Devices, have advanced into this project aiming to assess the feasibility of sensors in sports training, rather than to develop a marketable product.

The project was given a five week timetable for experiments and research work to be completed. Factors effecting the accelerometers used, such as climate and the presence of water, meant that the experiments with the rowing team had to be conducted indoors. The climate conditions of November, meant that it was too dangerous to actually conduct the experiments outdoors. Being outdoors also meant a limitation on laptop battery and would put the accelerometers, laptop and data acquisition boards at major damage risks, caused by the water splashes. Therefore, an indoor rowing tank

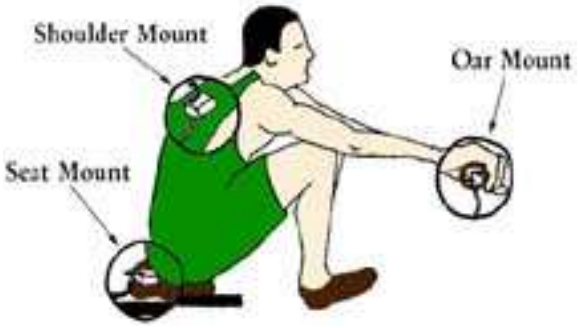

Figure 2.1: *Accelerometer placement in experiments*

was developed, to act as the platform for the boat.

Rowing was picked over other sports for a few reasons. Firstly, because of the repetitive body movements of the technique. Secondly, because of the relationship between the accuracy of technique and the resulting performance. And thirdly, because the movements for rowing is less random than other sports.

For the experiments, the rowers were equipped with three accelerometers. One mounted on the oar, one on the seat and one on the shoulder mount, as shown in figure 2.1. The sensors were set-up to produce thirty-two readings per second and between 75 - 100 readings were deemed enough to capture one full rowing stroke. A series of experiments were performed with the rowers, where they would each produce between 25 - 30 strokes per minute, applying different techniques, such as feathered and non-feathered strokes and faster slide and exaggerated motions. The logged dataset was then analysed to identify relationships for the different techniques of rowing. Much of the data was analysed in forms of graphs and tables, and helped to devise new training methods useful for coaches and athletes.

Their future work includes developing a user interface, which presents the data in

more meaningful terms to coaches and athletes. Other work includes the development of a device worn by the rowers, which would log data in real-time and provide feedback on the techniques.

The project concluded that sensors for sports training can be used to visually analyse data for repetitive motions, and provide information on avoiding bad techniques.

2.3 Hierarchical Recognition of Intentional Human Gestures for Sports Video Annotation

Gesture recognition has been performed by computer vision researchers but poses some performance limitations, such as lighting conditions, camera calibrations and computationally intense calculations, leading to ambiguity and mis-classification of recognised gestures. Hand sign language recognition [18] and T'ai Chi movements in virtual environments [2] are examples of computer vision based solutions.

The School of Computing from Curtin University of Technology in Perth, have been participating in the research of intentional human gestures and their correct classification [3, 4]. This research is done utilising sensors worn as wristbands, which could help sporting officials such as umpires and referees make crucial decisions in a match.

To facilitate this they set-up a video experiment, mimicking the action gestures of an umpire from cricket. The intention here is to create on-the-fly highlights for umpires and sporting officials, which they can bring up on the stadium screen through acting out various gestures, that correspond to specific events in the game. Timestamped video is continuously recorded and the movements/gestures of the umpire are recorded

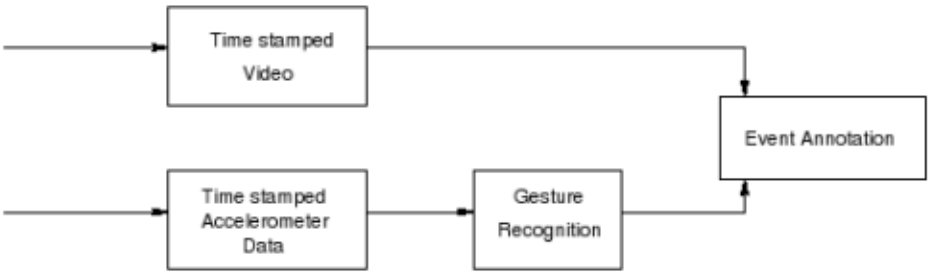

Figure 2.2: *System Architecture*

simultaneously. If a predefined gesture has been recognised and classed, an event is annotated corresponding to the video at the time of the gesture. These gestures are typically used by the umpires to signal their decision of play to the crowd and audience. Such gestures include signals for *Dead Ball*, *Four*, *Leg-bye* and a custom one such as *TV Replay*. For the experiment, a *TV Replay* gesture is predefined as drawing the outline of a rectangle in the air, with both hands. Figure 2.2 illustrates their system's architecture.

Clearly, this work outlines the huge range of applications that can be developed using sensors and way they can make our lives easier, through a more accustomed means of interaction.

2.4 Smart Artefacts

This section is subdivided and aims to give an overview of projects relating to sensor-augmented artefacts.

The vision of Ubiquitous and Pervasive computing includes smart artefacts, facilitating the development of smart spaces by supplying context-aware information. Much

research and work has been ongoing with smart artefacts, but this has given rise to the possibility of smart spaces, which are environments of interconnected "heterogeneous"[1] devices, communicating via some middleware. Work such as [10] and [13] are dedicated to providing solutions for integrating device inter-communication, within environments such as "home networks"[2]. Much of this context-awareness is born through the use of sensors, to make sense of the captured real-world data.

The Sensor-Augmented club is a sports artefact acquiring data from the real world. The next sections will discuss other sensor-augmented artefacts, which open up a huge range of possible applications.

2.4.1 Smart Couch

The Smart Couch [7] is an artefact utilising sensors, which was developed as part of the "CAMS"[3] project by the Distributed Systems Group, at Trinity College Dublin.

The CAMS project is research concerned with middleware for Ubiquitous computing. The Smart Couch is a smart artefact which, has the ability to sense and interpret data, making it context-aware. This project is an integral part of Ubiquitous and Pervasive computing, as it contributes greatly to the concept of having smart technologies in environments such as homes, faculties and offices.

The Couch itself has accurate sensors in the form of load cells, attached to each of its legs. The intention here is be able to recognise the person sitting on the couch

[1] heterogeneous devices are devices of different flavour in terms of hardware specifications and platform.

[2] home networks are intelligent homes, in which ubiquitous and embedded devices(appliances, heating and security systems) are plug and play, interconnected and communicating via some protocol.

[3] CAMS: Context-Aware Mobile Services.

given their weight. At the moment its prototype applications include:

- A Visualisation tool showing the activity of the sensors.
- A Speech generator, used to greet and interact with the person sitting on it.

This opens up the possibilities of interaction with an identified person, who might have an associated profile for his/her settings in the home environment or office. At the moment the couch can tell whom is using it and it also has ability to sense where on the couch the person is sitting. As a complete application, the smart couch is undecided, but it does demonstrate that it could be used for smart spaces or healthcare. In terms of healthcare, the couch could be useful for elderly, disabled people or even babies who are generally confined to a space, such as a bed or cot. This space could then be monitored to deliver tailored services.

The door has been left open for more applications, but the couch demonstrates that through the use of sensors a new pervasive service can be supplied, through a unique form of human computer interaction, paving the way for new technological advances for homes or healthcare facilities.

2.4.2 Smart sword

The Smart Sword [16] is a Japanese Shinai sword used in Kendo and it was also developed as part of the CAMS project. To demonstrate the usefulness of such a smart artefact, three applications for it were proposed.

- A light sabre sound effect generater.

- A Visualisation application, updating the movement of a 3-D Shinai model in real-time.
- A Training application intended on instructing moves for beginners.

These three applications utilise sensor data and could produce a very good training application if combined. As it stands, the smart sword demonstrates good uses for applications in games and sports training.

2.4.3 The Media Cup

The purpose of the Media Cup [8] project was to show that devices integrated with context-awareness, can offer improved performance and new functionalities. This allows them to be integrated into an environment, as non-computional artefacts finding themselves interacting via new methods.

The project team have developed a framework for artefacts augmented with sensors and context. The framework includes concepts such as:

- Autonomous awareness.
- Context-sharing.
- Context use.

Autonomous awareness in arefacts provides them with contextual information, used to assess their own state or situation. *Context sharing* artefacts are able to communicate data via some protocol, to entities within its region. *Context use* facilitates applications

Figure 2.3: *The Media Cup*

to use the context from these artefacts, for emerging greater functionality. Figure 2.3 shows the Media Cup embedded with sensors.

The design of the cup itself means that the technology is transparent to humans, as the peripherals are embedded in its base. This means that everyday artefacts, which seem normal, act normal also, whilst passively providing more functionality. The hardware is based on a circular board design and it includes a PIC16F84 microcontroller with 1MHz, 15k memory for the code, 384 bytes of RAM and a digital temperature sensor.

Its implementation is done in such a way that the device is not polled constantly, but the cup publishes an interrupt in the board once an event has been triggered.

Detected movements are recorded as an event and the history of these are logged to compute a rule-based heuristic, indicating context that relates to the use of the cup. Four movement contexts are defined as *cup is stationary, drinking out of cup, cup is played with* and *cup is carried around*. A temperature reading is also logged, to produce further contexts, such as *filled up, cooled off* and *current temperature*.

The Media Cup broadcasts its context with its unique IP every two seconds, using an infrared diode. Its' transmitting range is limited to two metres and the information is collected through overhead transceivers, installed in the environment. The transceivers are linked via a "CAN""[4] bus and connected also to a gateway on the "LAN"[5], where the contextual information is transported in the form of "UDP"[6] packets

2.5 Summary

In this section, I have presented my compiled research, which I believe are the broad applications born from sensory technologies. I have presented two sports training applications; *Application of Accelerometers in Sports Training* and the *Shinai Sword*. The first will provide analysis and feedback for coaches and athletes in Rowing, and the latter will instruct techniques pivotal in mastering Kendo. I have discussed *Hierarchical Recognition of Intentional Human Gestures for Sports Video Annotation*, which aims to bring about a new means of quick interaction for video annotations, provided with the use of sensors. Finally, I have mentioned smart artefacts, such as the *Smart Couch*

[4]The CAN serial communication protocol is a standard defined by ISO (ISO 11898) and its existence mainly came about for networked communication within the automotive industry, in 1986.

[5]Loca Area Network

[6]User Datagram Protocol

and the *Media Cup*, which gather and interpret sensor data as contextual information, in order to interact with their associated applications.

This state of the art review highlights the important role of sensors, in creating a broad range of applications and by facilitating a new means of human computer interaction, and by extending the functionality of ordinary everyday artefacts.

Chapter 3

System Design Overview

For the "Sensor-Augmented Golf Club", the sensor hardware was obtained from Xsens Motion Technologies[15], who are based in the Netherlands. This chapter will provide a design overview of the system, whilst providing details on each hardware component.

3.1 The Mtx Inertial Sensor

The Mtx is a minature inertial sensor with an embedded processor, which has the ability to output 3D orientation, 3D gyro(rate of turn) and 3D magnometer data in real-time. Figure 3.1 shows the mtx.

The Mtx runs a sensor fusion algorithm, in order to calculate its orientation and movement in three-dimensional space. The algorithm is said to combine the three mentioned types of data, to act as an "attitude and heading reference system". Typically, you can find an AHRS onboard many aircrafts, in which the 3D orientation, 3D gyro and 3D magnometer for each axis (X,Y,Z) is calculated, and a Kalman filter is applied

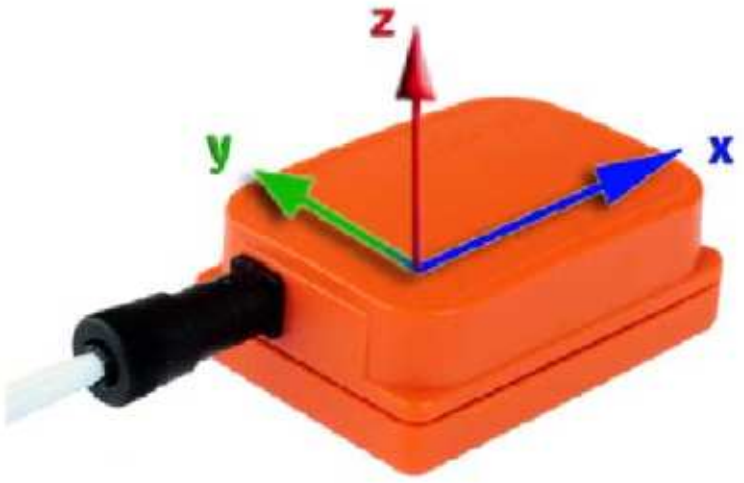

Figure 3.1: *The Mtx from Xsens*

to compute the solution from these multiple sources. Figure 3.2 shows how the fusion algorithm works.

3.1.1 Mtx Output Modes

The Mtx is capable of producing two timestamped output formats, simultaneously.

- Calibrated data.
- Orientation data.

The *Calibrated* sensor readings are in the right handed Cartesian co-ordinate system, as shown by the X,Y and Z axes in figure 3.1. This co-ordinate system is fixed to the Mtx's housing, regardless of its orientation. The output units of the calibrated data is as follows:

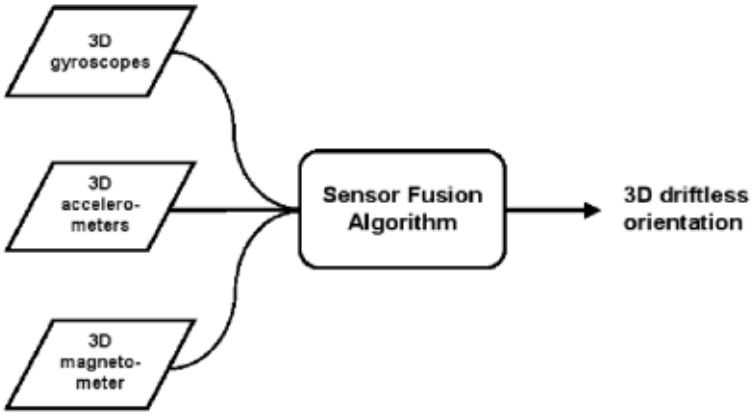

Figure 3.2: *Sensor Fusion*

- Acceleration, in m/s^2.
- Angular velocity(rate of turn), in rad/s.
- Magnetic field, in a.u("arbitrary units"[1]).
- An optional output of temperature, in degrees.

There is also the possibility of receiving data, in *Uncalibrated* format from the 16-bit AD-converter embedded in the Mtx, which means data will be raw and unprocessed. This type of data will not be useful for my application.

The *Orientation* is calculated between the sensor fixed co-ordinate system S and an earth fixed reference co-ordinate system G. The earth-fixed co-ordinate system also adheres to the right handed Cartesian co-ordinate system.

The orientation is calculated using the sensor fixed co-ordinate system S against the earth-fixed co-ordinate system G, whilst using G as the reference co-ordinate system.

[1]normalised to the earth field strength.

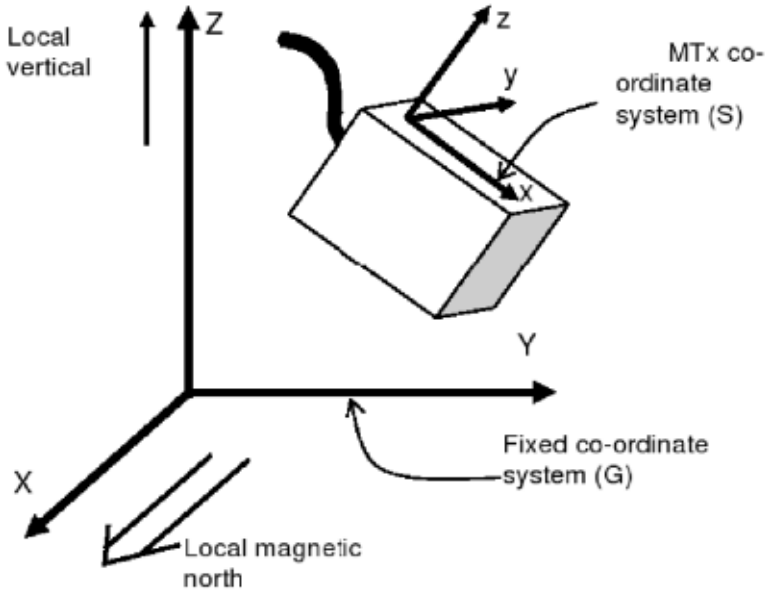

Figure 3.3: *Co-ordinate systems*

Figure 3.3 shows these right-handed co-ordinate systems at work.

From Figure 3.3, you can deduce that:

- X readings are positive when pointed to the local magnetic North.
- Y complies with the right handed co-ordinates facing West.
- Z is positive when facing up.

The orientation output can have three sub-formats:

1. Unit Quaternions, also known as Euler parameters.
2. "Euler angles"[2] in the form of roll, pitch and yaw.
3. Rotation matrix(directional cosine matrix).

[2]Euler angles output was the selected orientation output mode for my application. Roll is a rotation

3.1.2 Performance limitations

The Mtx is quite vulnerable to objects containing ferromagnetic materials or objects which themselves are magnetic. In the case of these materials i.e iron, engaging in close proximity to the Mtx, the *Yaw* readings from the orientation mode, will appear corrupt. Likewise, large bodies such as vehicles passing nearby, have been known to distort the calibrated output readings.

Also, if the Mtx is subjected to vibrations, this saturates the accelerometer, which leads to further corrupt readings of the *roll* and *pitch*.

In order to solve the ferromagnetic issue, the environment must be carefully chosen, or the nearby ferromagnetic object must be "degaussed"[3].

3.2 The Xbus Master

The "Xbus Master"[4] acts as a hub for up to ten Mtxs connected via a daisy chain method. It supports two daisy chains, each capable of containing ten Mtxs, and these are connected via an Xbus cable. The XM has the ability to facilitate power to the small Mtx sensors, whilst extracting the synchronously sampled sensor data at frequencies up to 51Hz . It is a battery powered unit(an ac adapter is also supplied) delivering up to three hours of uptime. In terms of remote connectivity to a host, it is equipped with

around the X axis defined from -180° to 180°. The Pitch is a rotation around the Y axis defined from -90° to 90°. The Yaw is a rotation around the Z axis defined from -180° to 180°. All positive rotations are right-handed.

[3]Degaussing is a process of reducing the magnetic field effect. This is done by applying strong alternating magnetic fields with decreasing magnitude in random directions to the magnetised object.

[4]Will be abbreviated as XM from here onwards.

Figure 3.4: *The Xbus Master*

a USB-RS232 cable, but also provides wireless connectivity via Bluetooth. Figure 3.4 shows the XM.

The XM is used in my application to connect to two Mtxs. One Mtx is augmented with the golf club and the other is worn on the shoulder. The XM itself is worn around the waist using a belt.

3.2.1 XM states

The XM can be in four different states. The temporary states are *Wakeup* and *AutoStart*, and the main states are *Config* and *Measurement*. Whenever the XM enters a temporary state, it automatically switches to another state. Figure 3.5 shows the

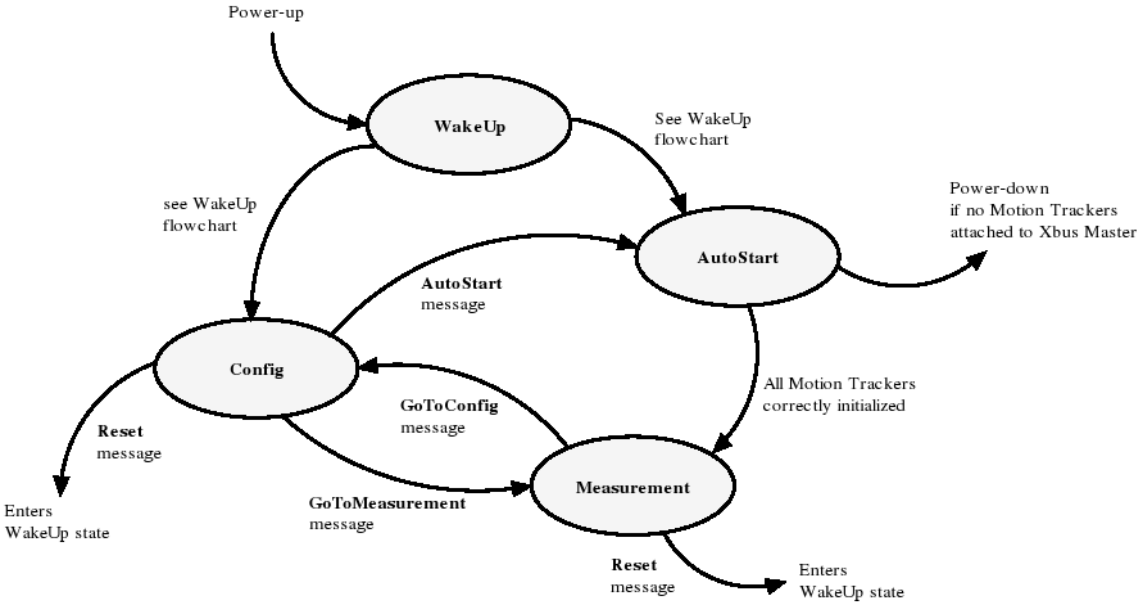

Figure 3.5: *The XM state flow chart*

XM's states.

The Wakeup state

The XM powers up and finds itself in the temporary *WakeUp* state, and depending on the actions taken here, it will move to one of the two main states as shown in the state flow diagram.

The XM sends a **WakeUp** message to the host and will enter the *Config* state once a **WakeUpAck** is received within 500ms. If this does not happen, it transitions to the *AutoStart* state, where it will scan its bus to initialise the connected Mtx motion trackers. If no sensors are found the XM powers down. After these actions in the *AutoStart* state it proceeds to the *measurement* state.

The procedure is slightly different when in the *WakeUp* state and connected via bluetooth. It begins as previously mentioned, by sending a **WakeUp** message to the host via serial interface and if serial interface is to be used, then a reply of a **WakeUpAck** within 500ms should set this up and then the XM enters the *Config* state. If this reply is not received by the XM, it will check the **DisableBluetooth** setting. If this is set to one, Bluetooth initialisation is skipped and the XM jumps to the *AutoStart* state utilising the serial connection. If the setting is zero, Bluetooth is enabled and the XM tries to connect to the host via Bluetooth. If this fails the XM enters the *Config* state utilising Bluetooth. The remote host can now search for the XM and connect to its Bluetooth serial port and then stay in *Config* state.

If the XM successfully makes a connection to the remote host, it sends the **WakeUpAck** using the Bluetooth Serial Port Profile. When the XM receives the **WakeUpAck** within 500ms, it enters the *Config* state or otherwise the *AutoStart* state. Figure 3.6 shows this process.

The AutoStart state

The transition to this state happens when the XM fails to receive a **WakeUpAck** message in time from the host, but it will also transition to this state when it does receive an **AutoStart** message from the host.

The main function of this state is to reset and an scan the XM's bus, whilst aggregating all the attached sensor ID's in ascending order, starting from the lowest device ID. They are then assigned Bus ID's(BID's). Once the XM has delegated BID's to each Mtx, it sets itself and the Mtx's to *Measurement* state.

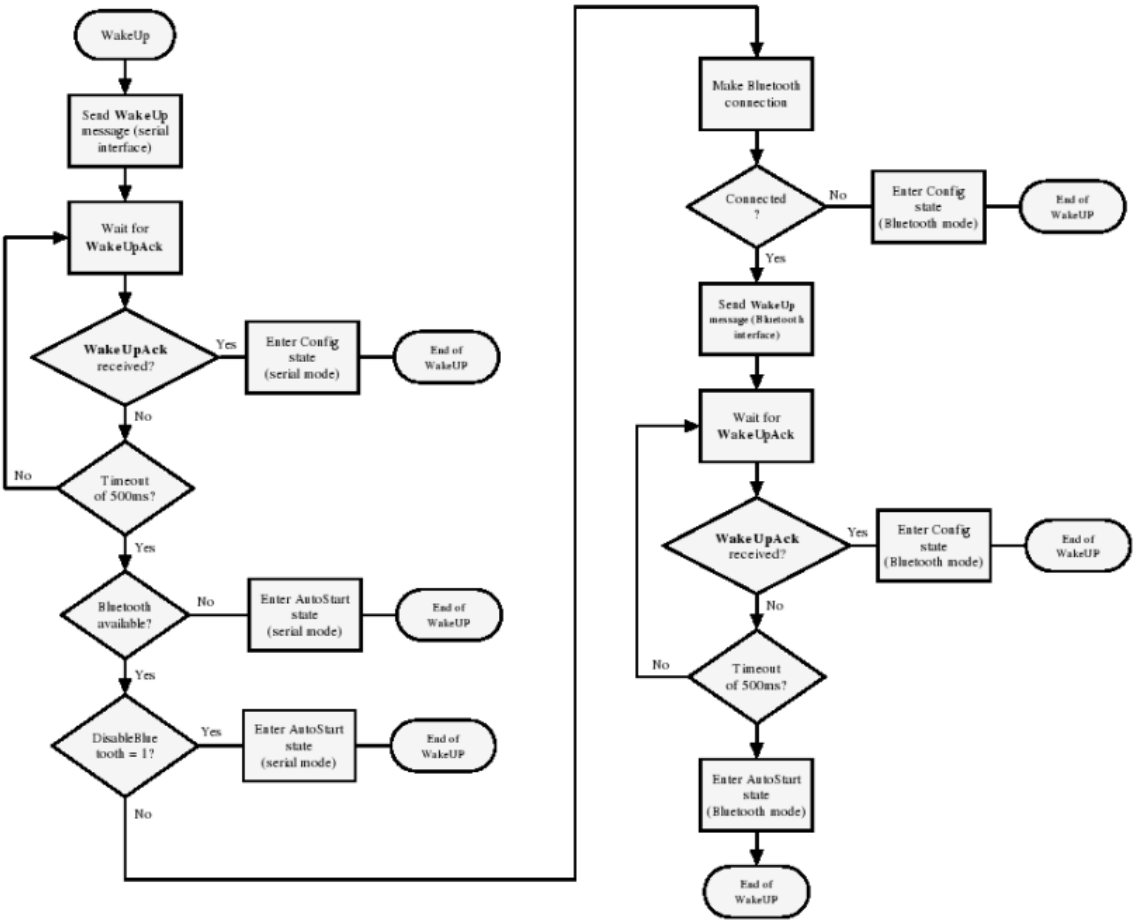

Figure 3.6: *The Bluetooth* ***WakeUp*** *flow.*

The Config state

This state is useful in order to send messages to the XM, to tweak its various settings, such as **SetPeriod** which sets the sampling period.

The Measurement state

The initialisation of this state first requires a test run, where the XM instructs the Mtx's to start sampling data. The Mtxs' are polled one by one and they send their data to the XM. The XM then checks if the correct sample frequency is set, in order to obtain the sensor data. If there are any problems, the XM sends a "not-acknowledged" message with the correct error code. If on the other hand the test run is successful, the XM sends an "acknowledge" message.

Now the measurement begins and the data is transmitted to the host with a **BusData** message, for every sample instance.

3.2.2 Performance limitations

I noticed that issues generally arise when operating via Bluetooth. The XM is said to have the capability of operating over a 100m range and a line-of-sight is often needed, to avoid data throughput problems.

Bluetooth uses microwave frequencies and the presence of water seems to have a negative effect on transmissions. Microwave ovens in the region also cause a loss or a slow delivery of data.

Other Bluetooth devices interfere with packet transmission and I noticed this when

having bluetooth switch on my mobile phone. The presence of strong WiFi signals also caused my application to suffer a 2-3% packet loss, for every 50 readings per second.

3.3 HP nx8220 Laptop

A HP nx8220 laptop was used as the host machine running my application which records, stores and analyses the data accumulated by the XM. A *Serial Port Profile* was created in order to pair the XM with the laptop. Some of the relevant features of the laptop include:

- 2Ghz Intel Centrino processor.
- 1Gb memory.
- Bluetooth module.

The application was developed on Windows XP professional and coded in C++, using an "IDE"[5] called *VisualStudio.net*.

3.3.1 Performance limitations

In order to successfully retrieve data packets via the bluetooth option, it was imperative that the integrated wireless card remain disabled, or unconnected to any WLAN's, in order to ensure that the sampled fifty readings per second were produced without skipping timestamps(i.e packet loss). More detail on this will follow in later chapters.

[5]Integrated Development Environment

Figure 3.7: *The Sensor-Augmented Club*

3.4 The Sensor-Augmented Club

As previously mentioned, the Sensor-Augmented club is based on a one wood driver. The driver used for the project is particularly old but still in good condition. To attach the sensor, the clubhead was filed down to make it more flat, in order to adequately support the sensor, which was attached using double sided tape. A level was used during the filing process, to make sure the sensor was completely level on the clubface, thereby removing any bias. Figure 3.7 depicts the Sensor-Augmented club.

Figure 3.8: *The Hardware Design View*

3.5 Top-Level Design View

This section outlines a Top-level view of the connected components, which were individually mentioned earlier. The user is equipped with the XM strapped to his/her waist. The two Mtxs' are connected to the XM via Xbus cable. One of these Mtxs' are placed on the shoulder and the other is appended to the clubface. At this point the XM is reading in the data from both Mtxs' and re-arranging it by ascending device ID, starting with the highest ID. The XM then transmits this data as data packets to the laptop via bluetooth, whilst the application is running on the laptop. This data is then ready for storage and analysis, by the application. Figure 3.8 shows the design overview.

Chapter 4

Swing Analysis

4.1 Introduction

In this chapter, I will present details on swing analysis. The intention here is to provide an understanding of the different components of a golf swing, and the techniques required to deliver a perfect swing. I will also be discussing the experimentation process undertook, in order to capture the dataset produced by the sensor-augmented golf club. Using this data, I could then train my application to recognise the different parts of a golf swing, whilst verifying the technique used.

4.2 Components of a Golf Swing

I mentioned in the first chapter, that one is permitted up to fourteen clubs in order to take shots throughout the course. These clubs include mainly irons and woods, and whilst the technique varies slightly from the irons to the woods, the principle

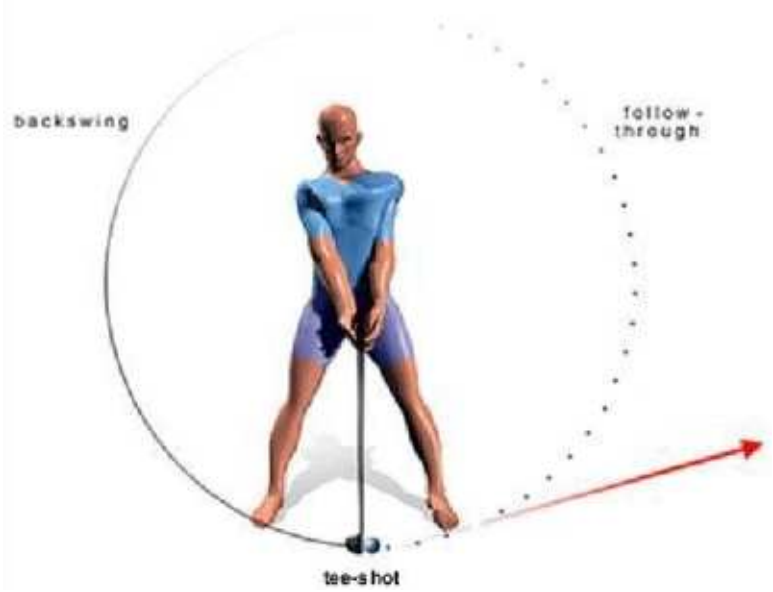

Figure 4.1: *The Components of a golf swing*

components of the golf swing remain the same. Figure 4.1 illustrates the components of a golf swing. Golf is a sport which commands absolute technique and symmetrical movements. The one wood is a famous club, used to tee off in order to drive the ball as far as possible. Tiger Woods is a professional golfer who is notorious for the longest drive(300 yards) and for the precision of his technique, which allows him to deliver more power at club-ball impact. Only by a substantial amount of practice over numerous years, can you master this technique and put your swing through a 360° rotation. For many amateurs the swing trajectory is more like two-thirds of that. From figure 4.1 you can see that there are three different parts of the swing.

To use a driver, you must have a wide positional stance, by placing your feet a shoulder width apart. You must also bend the knees slightly, keep the back straight and the head down with your eyes fixed on the ball. Your posture should be not ridged,

but ready for movement, and sitting back in your stance helps this.

A rotary movement of the shoulder and upper body is required, whilst your legs remain in tact. Many professionals train their body through various exercise routines, which help the upper body wind, and un-wind throughout the swing. It is an unnatural movement, having to keep your legs in place, whilst you rotate only the upper torso and shoulders while, also keeping the head down. Ball placement should be in-line with the tee, and you must align the driver accordingly.

During the *backswing*, the aim is to bring your left shoulder across the midpoint of your shoulder-width stance, while keeping the left arm straight and close to the body. This creates the arc of a circle with the clubface, and you will follow the exact path of this arc back through to the tee-shot. For most amateurs it is enough to bring the club across the midpoint of your stance, but most professionals have the ability to bring it over to their right foot. This leads to a higher arc on the backswing, which greatly contributes to having more power when unwinding. My application will be used for training amateurs and semi-professionals, so the height of the *backswing* and *follow-through* will be limited.

As you unwind your *backswing*, you follow the same path in order to bring the clubface back to its original position, which was square at the tee. The aim is to train yourself to *follow-through* the *tee-shot*, rather than stopping once the ball is hit. This technique provides greater impact on the "sweet spot" of the clubface. When a golfer avoids this technique, unconsciously he/she thinks the job is complete at the tee-shot, this can lead to snapping of the wrists, which will slow the clubhead speed and provide a curved hook, on the flight of the ball [9]. Acquiring the correct technique is the

holy grail for amateur golfer and more power will be delivered eventually through the reptition and remembrance of proper swings.

4.3 Physics at Work

A golf swing has physics written all over it, with a great example of angular motion. The twisting swing produces torque on the club. The torque changes the angular velocity of the club causing a rotation. A key element in producing long drives, is the velocity of the clubhead, which is determined by the speed of unwinding the upper body. This leads to a faster speed of the clubhead at the tee-shot and the amount of kinetic energy transferred from the clubhead to the ball, is proportional to the mass of the club and the square of its velocity. Professional golfers produce clubhead speeds of 160 km/ph but Tiger Woods is the only golfer who can deliver a club speed of 201 km/ph, on the downswing. As a result the ball will be travelling at 290 km/ph two feet after the tee-shot, that is 32 km/ph faster than the average tour professional. Research has also shown that an optimum lie angle will maximise the throughput of the swing [11]. A lie-angle of 7.5° allows Tiger Woods to achieve longer drives.

A longer and more successful drive depends on some factors. The club speed is determined by the effectiveness of the swing technique and velocity of the ball is determined by the club speed. When the ball is hit by the club, it deforms slightly, this is due to the elastic core of the ball. Energy is then transferred and stored from the club to the ball and it tries to retain its original shape again. Good impact on the ball causes a spin of up to 50 revolutions per second and this spin greatly contributes to a

longer flight of the ball. This spin is exaggerated because of the golf ball dimples and aerodynamics known as "Magnus effect"[1], which help propel the ball longer as it curves in flight [6, 12]. This Magnus effect occurs due to the drag force on the air at the top and the bottom of the ball. The air tends to mover quicker on the top as opposed to under the ball, working on the dimples to create a lifting effect. Of course, the success of a drive is also determined by wind factors present, which can be random.

4.4 Data for the Swing

In the last chapter, I gave a system overview and introduced the Xsens hardware. The XM came with a software development kit, which included the MT software. I used this "MT SDK"[2] to capture the dataset from the sensor-augmented club.

The MT software shows the orientation of the Mtx in real-time and it logs the orientation data in the output mode of your choice. In my case I chose to have Euler angle output for the orientation, and Calibrated data output was also selected to provide the linear accelerations in the X,Y and Z axes relative to the Mtx. The MT software can handle input from pre-recorded files or real-time input. As I required real-time input, I first set the XM to run in bluetooth mode using the MT software's "Xbus Master settings" option. After this, I simply needed to tell the software to sample at 50 readings per second, also which COM port the bluetooth SPP was using, and finally I enabled the *timestamp output* option. Figure 4.2 shows the MT software.

[1]The Magnus effect is a peculiar lifting effect experienced by rotating bodies through a medium.
[2]Motion Tracker Software Development Kit

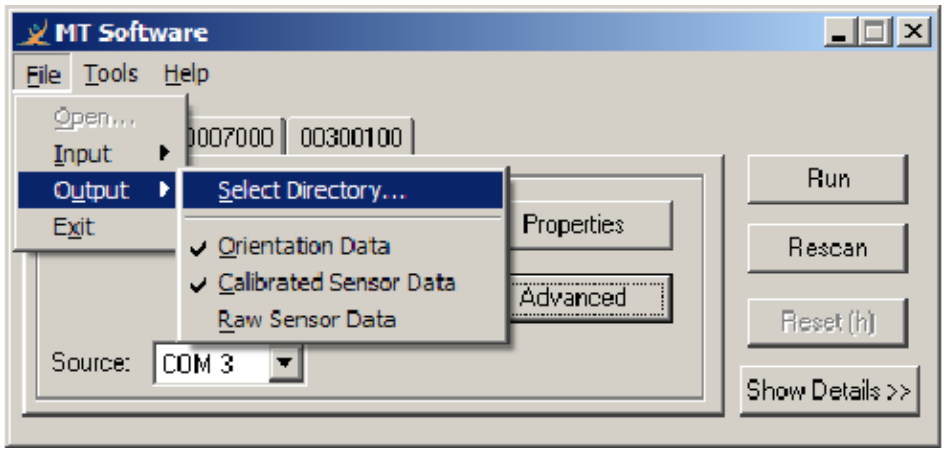

Figure 4.2: *The MT software*

The MT software logs all of the data transmitted by the XM in ASCII format log files. The Euler angles orientation data is logged to a file called "MT_euler_DID_XXX.log" and the Calibrated data is logged to a file called "MT_cal_DID_XXX.log". The log file naming convention is as follows. The DID denotes the Mtx device ID and the "XXX" is just a unique incremental number. Figure 4.3 shows a typical calibrated logfile. For ease of understanding, I have entered the column headings manually.

4.5 Lessons & Experiments

In order to prepare myself for experiments and gain a better understanding of the driver, I decided to take lessons twice a week for a month, at the local driving range. I had some experience playing "pitch and putt" and also par three golf prior to this, but I had never actually used the one wood many times before.

After this period, my tutor and I conducted some experiments for logging the

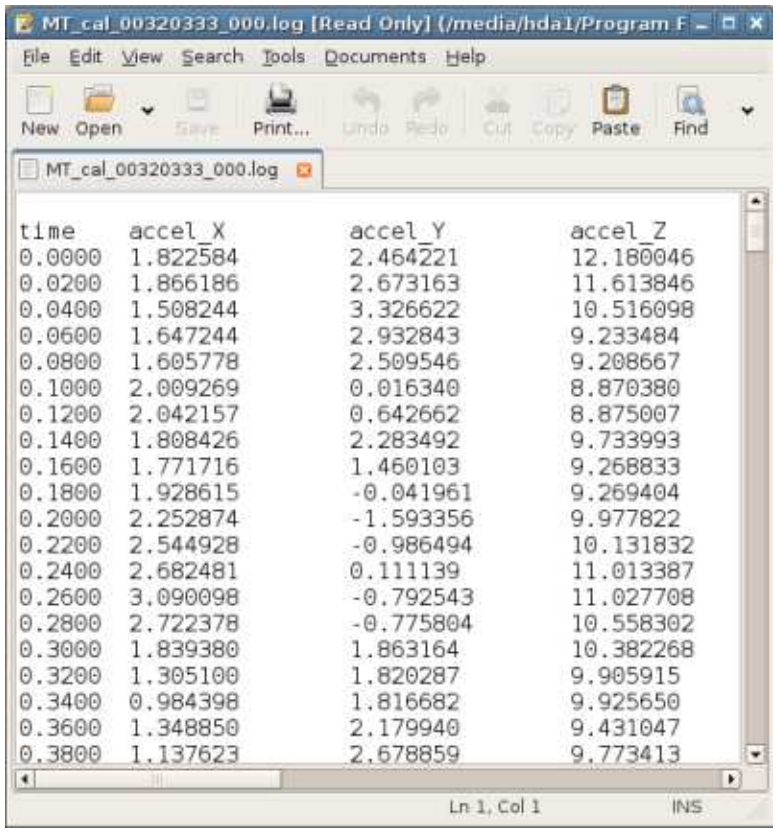

Figure 4.3: *A Calibrated logfile*

dataset produced by the sensor-augmented club. With the XM worn on a belt and an additional sensor attached to the rotating shoulder, I advised my tutor to take part in nine timed tee-shots using the Sensor-Augmented club. A *heading reset* was done when the club was aligned with the ball and the user was positionally sound, just before the shot. The intention here was to reset all the orientation values to zero, respresenting the club in a static and square(default) position.

All of this data was saved on the HP laptop and the bluetooth connectivity meant a reduction in wires, thereby allowing a free motion for the swing. The result of his experiments showed that the longest time it took for a complete drive was t=2.1 seconds and the shortest drive was t=1.8 seconds, where *"t"* is equal to time.

After the tutor, it was my turn to have some of my swings recorded. This was done to see the differences in sensor readings for a semi-professional versus an amateur. Armed with the data log files, I was now ready to analyse the data and see if I could draw relationships between numbers in the form of floats, and movements of the club.

From analysing the orientation log files, I could interpret that the sensor readings which would be relevant for my application were as follows:

- Euler angle floats for the Roll(ϕ), Pitch(θ) and Yaw(Ψ).

From analysing the calibrated log files, I could interpret that the sensor readings which would be relevant for my application were as follows:

- Acceleration floats for the Y and Z axes.

The Euler angles would give me the clubface orientation at all times, whilst the acceleration data would indicate movements in a particular direction. I chose not to

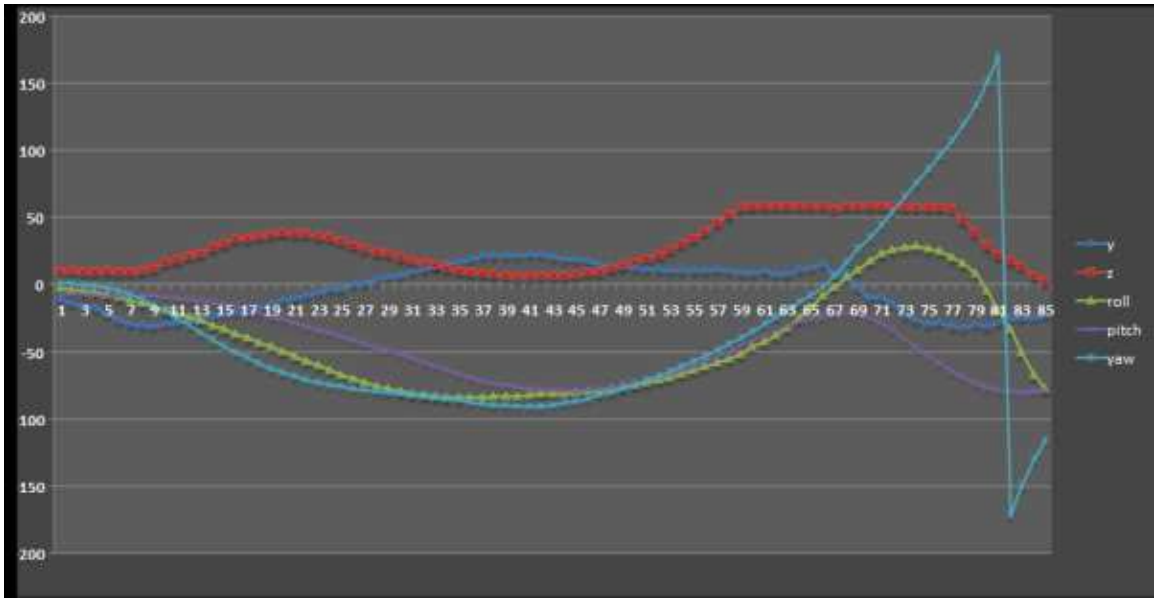

Figure 4.4: *A graphed representation of orientation and calibrated data*

use the *rate of turn* data or the *magnetic field* data from the calibrated log files, as the figures didn't fluctuate in accordance with any movements of the club.

4.6 Graphed representations of data

In order to study the log file data in an easier manner, I graphed the nine trials of my tutor in dotted line graphs. Visually it was now clearer to assess what was happening. Figure 4.4 shows a visual respresentation of the euler angle data and the accelerations in the Y and Z axes, for the lifecycle of the swing. This was directly graphed from one of my tutors swings and the data for 0-85 timestamps is depicted in the graph. The calibrated data of interest is the acceleration data(m/s^2) in the Y and Z axis. During the backswing, there is a negative acceleration in the Y axis, and as the club is raised

nearer the top of the backswing, there is a positive acceleration in the Z(vertical) axis. When static there is generally a positive acceleration of 9.81 m/s^2 in the Z axis, but this is largely due to the constant G-force applied to the Mtx by the acceleration due to the earths gravitational pull.

If you look closely at the graph, you will notice that on the first timestamp(1), the acceleration in the Y and Z axes is already -10 and +10 respectively. This is because at this point the swing was already underway and an event trigger had to be specified in my application to recognise this. More information on that will follow in the next chapter. Therefore, the graphed lifecycle is missing approximately fifteen timestamps, which if added to the eighty-five shown, brings the lifecycle to about two seconds in total.

With the sensors, whenever there is acceleration in a given axis followed by a halt, the readings slow down until they are gradually followed a large number of negative readings and vice versa.

If you look at the darker blue line, which represents the Y axis, you can see that initially it experiences a negative acceleration until about 25 timestamps later(1/2 second), at that point the club has reached its limit on the backwing. You can then see a positive acceleration in the Y axis, illustrating the club unwinding on the downswing, until it reaches a lower acceleration again. At this point the acceleration in the Z axis(red line) increases dramatically, this is because at this point the orientation of the clubface is accelerating upwards in the Z axis, denoting the follow-through.

In a similar way, I have interpreted the Euler angle dataset. The light blue dotted line represents the yaw, which initially is reset to zero before the swing, but as you

bring the club backwards for the backswing, the yaw almost reaches a -100 reading. The green dotted line represents the roll, and you can see that it too experiences a dip of almost -100 for the backswing. The roll and the yaw start to become positive at t=47 onwards. This illustrates the club coming back, following the same path for the downswing and they will both continue to grow positive through the tee-shot. The tee-shot can be recognised from the graph, when both roll and yaw cross over zero from being negative. These two readings occur close together at t= 66 and this indicates that the club orientation has become square at the tee-shot position again.

The shortest part of the swing is the follow-through, from t=67 onwards. At this point the orientation of the club will mean positive readings for the roll and yaw, until the sensor is turned upside and the readings both plummet.

In this way, I have trained my application to recognise the gesture of a swing and the next section shows more details on this and how I verify the club's orientation and also the shoulder positioning.

Chapter 5

Implementation

In this chapter, I will present implementation details of my C++ application. Firstly I will give an overview of how communication to the sensors is possible and from there on, I will present details of how the program control structure works.

5.1 Interfacing through the MT Object

The MT Object provides a COM[1]-object API which facilitates a new abstraction layer between the Mtx and my software application. Through a file called "MTObj.dll", I can access many functions of the sensors, in order to obtain the data, or to tweak its settings. This COM object facilitates hardware communication interfacing and real-time performance. The MT-Object allowed me to also access the sensor data through applications such as MS Excel and Matlab, in order to visually assess the sensor readings.

[1]Component Object Model

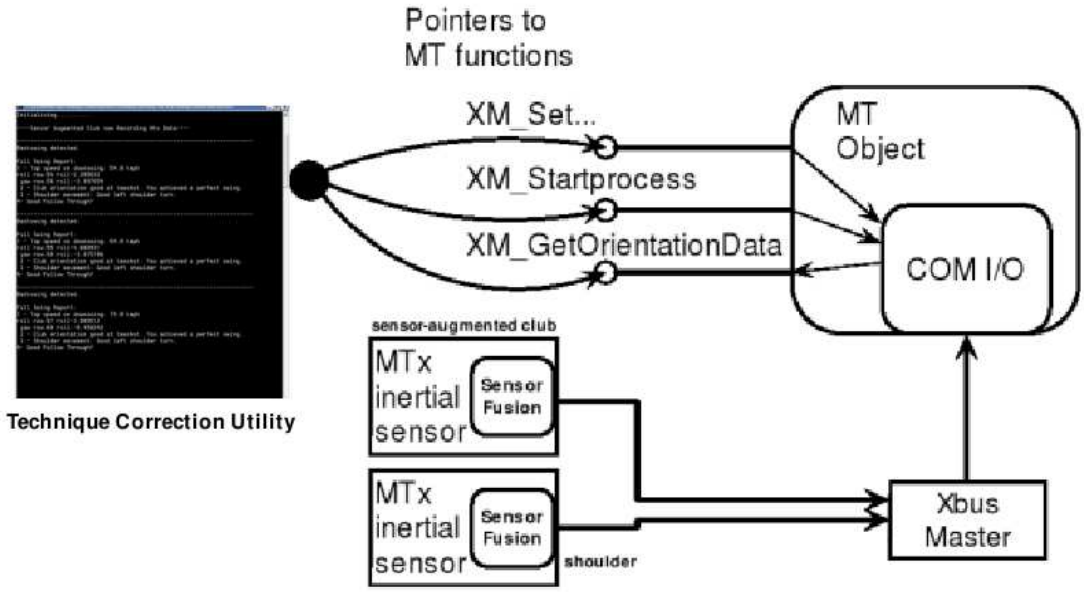

Figure 5.1: *Data flow when using the MT Object with the XM*

The MT(Motion Tracker) object allows accessibility to the Mtx's and the XM, with specific function calls. These sets of functions were incorporated into my application and the purpose of these are to give software developers more time to concentrate on the implementations of their algorithms, rather than spending time developing functions which retrieve or configure the sensors in any way. Figure 5.1 shows the data flow when using the MT Object with the Xbus Master. From figure 5.1 you can see that the XM collects and aggregates the sensor data. My training application, which I have named the "Technique Correction Utility", calls functions which configure and retrieve the sensor data, and this is delivered through the COM object. From here, my application is ready to work with the relevant orientation and calibrated data from both sensors.

5.2 COM

COM[1] is a software component standard referring to a specification and implementation developed by Microsoft, which provides a framework for integrating components. The framework facilitates interoperability and reusuability of distributed software components. It allows developers to build systems by assembling reuseable components communicating via COM. COM defines an *application programming interface*(API) which allows integration between custom applications and which allows diverse components to interact.

Interaction is facilitated when components stick to a binary structure. When this is done, components written in different languages can interoperate.

COM components contain executable code distributed as Win32 dynamic link library's(DLLs) or executables(EXEs). When these are registered in the windows registry, the COM library can use this to get the location of a DLL or EXE.

Some advantages of COM include:

1. Wire Level Standard: Users of components do not have to implement or handle underlying network mechanisms such as *TCP/IP* or *Serial communications* in order to use the components. These layers can remain transparent.

2. Binary Standard: Communication between server or client components which are developed in different languages is possible once using the binary standard.

3. Runtime Polymorphism: at runtime, clients can detect the components it needs to use for services.

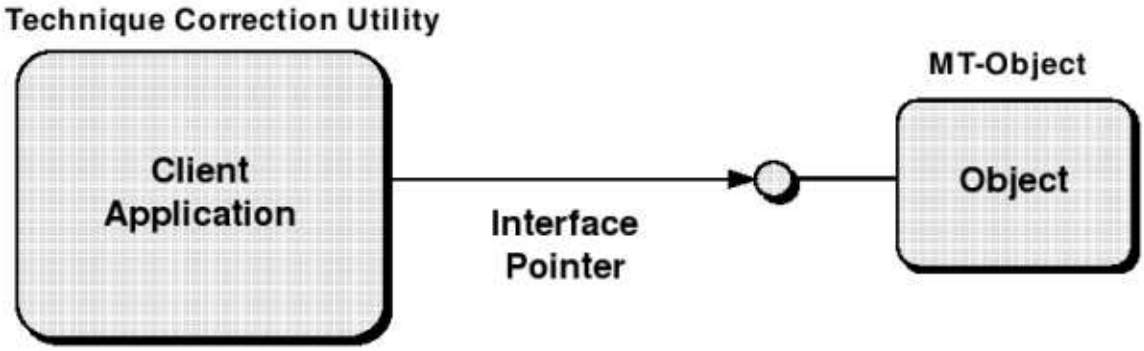

Figure 5.2: *The interface extending towards the client application*

5.2.1 Objects and Interfaces

An object is an instance of some *class*. A class contains variables and functions which can provide some service to its clients. An object that adheres to the COM specification is much like a C++ object, except that the object cannot be completely accessed directly and clients must access it through a clearly defined contract known as an *Interface*.

An *Interface* is a strongly-typed group of semantically related functions also known as "interface member function". In COM, all interface names are prefixed with an "I". Also, a GUID[2] is allocated to interface and the COM system uses the GUID when operating on interfaces. The interface defines how it should be used and what behaviour is expected from an object through that interface. Interfaces are accessed by its pointer. Figure 5.2 shows the communication between my application and the MT-Object facilitated by the "MTObj.dll" file.

[2]Globally Unique Identifier

5.2.2 COM data Retrieval

Both Orientation and Calibrated data comes from the MT Object. The data is returned in arrays of type *float*, representing degrees. The Euler angles data sample, will contain three elements, so for the *fOrientationData* array, index [0] = roll, [1] = pitch and [2] = yaw, respectively. If there is more data available from other sensors, i.e the shoulder Mtx, this data is concatenated to the end of this array. The array size should double and the data is returned, starting with the device which has the highest ID. Timestamped output, when enabled will pre-pend the timestamp at the beginning of the array. So [0] now becomes a timestamp of type *float* and [1] now becomes the roll value.

The same procedure is true for the calibrated data readings except that a typical calibrated sample consists of ten values(eleven when timestamp output is enabled). The *fCalibratedData* array is also of type *float* and is illustrated in listing 5.1.

```
fCalibratedData[0] = accX [m/s2]
fCalibratedData[1] = accY [m/s2]
fCalibratedData[2] = accZ [m/s2]
fCalibratedData[3] = gyrX [rad/s]
fCalibratedData[4] = gyrY [rad/s]
fCalibratedData[5] = gyrZ [rad/s]
fCalibratedData[6] = magX [a.u.]
fCalibratedData[7] = magY [a.u.]
fCalibratedData[8] = magZ [a.u.]
fCalibratedData[9] = temp [C]
```

Listing 5.1: *An fCalibratedData Array*

The first three values in the array represent the linear acceleration in the X,Y and Z axes, respectively. The following three represent the angular velocity(rate of turn) around the X,Y and Z axes, respectively. The next three denote the magnetic field in the X,Y and Z axes, respectively. There is also a temperature measurement appended to the end of the array. Similar to the orientation readings, if there is more calibrated readings available from another sensor i.e the shoulder Mtx, then this data is concatenated to the array and the data is sorted in order of the highest Mtx device ID.

5.3 Event based communication

Event based communication is incorporated in my application, as opposed to continuously polling the sensors. In polling based communication, the application queries the sensors periodically when only the latest sensor readings are relevant. In this way, the query function is applied to run a loop and query the MT Object at intervals, when the latest orientation or calibration data is required, instead of retrieving all the data samples.

In my application, I have implemented event based communication from the XM to the application. The XM continuously collects data from both sensors and temporarily stores these values into two buffers [3]. The producer-consumer model is applied and

[3]Buffers exist in the form of *safearrays*. There is one for calibrated data and another for orientation

the produced data is placed into this buffer. If this data is relevant, my application consumes the data for further processing. The buffer is constantly filled and then cleared before new values are placed into it again. The rate at which the buffers are handled, is determined by the data sample rate.

When a potential gesture of a swing is recognised, this acts as a trigger, and my application will start retrieving and saving all of the calibrated and orientation data from both sensors. This data can be processed later. In figure 5.3 you can see that both orientation and calibrated buffers are being continuously filled and cleared, in a loop. Once the trigger condition has been satisfied, the program continues to save and process the data. Finally, the Technique Correction Utility presents feedback to the user. More detailed information will be presented later.

5.4 Communication Timing

When the Xbus Master is in its measurement state, it sets the Mtx's to be in measurement state also. Figure 5.4 shows a flow of processes which contribute to the timing between data acquisition and data output. The internal DSP[4] continuously loops as shown in figure 5.4. The trigger itself is specified in the application and will be discussed later. The time delay between calculating a new sensor reading, is determined by two factors:

1. Internal acquisition and calculation time.

data. *Safearrays* keep track of their limits and bounds carefully and are encapsulated as variants before being passed between different COM objects.

[4]Digital Signal Processor

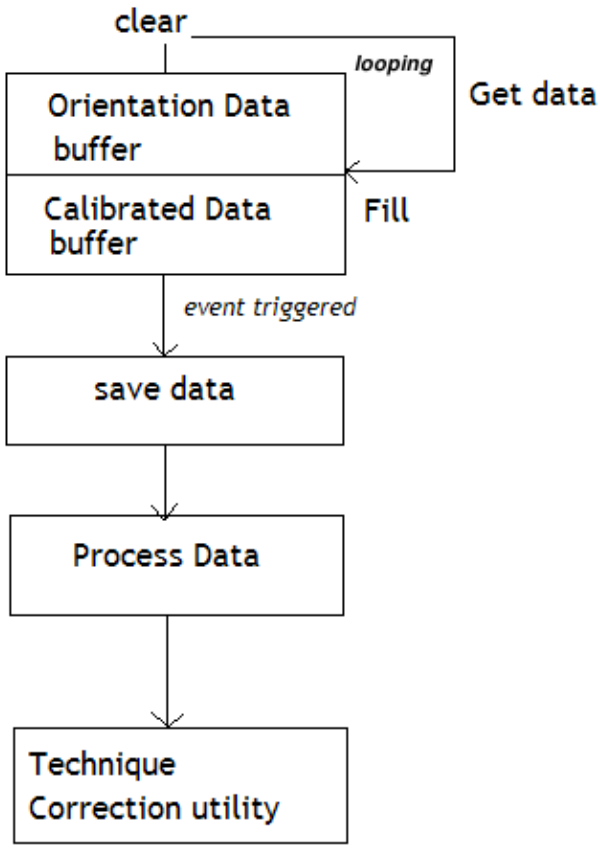

Figure 5.3: *Event Trigger*

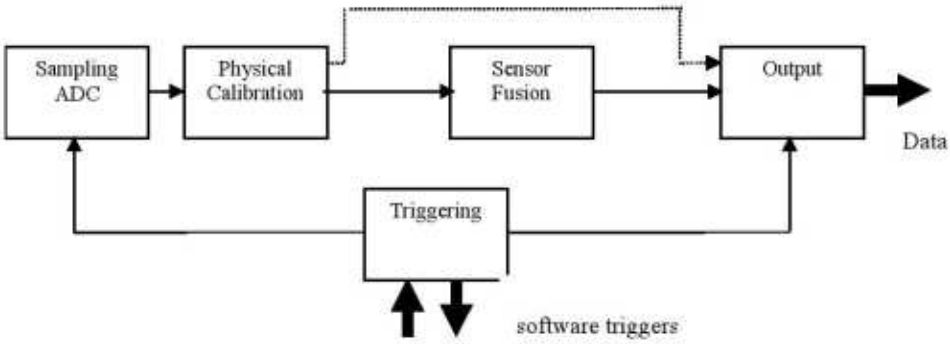

Figure 5.4: *The digital signal processing loop*

Total bytes in message * 10 bits/byte
---------- = transmission time in seconds
communication baudrate (bits per second)

2. Bluetooth transmission time.

The internal acquisition and calculation time for Orientation output is 6.43 ms and for Calibrated output is 1.08 ms.

The following formula helps deduce the Bluetooth transmission time:

The euler angles(3 floats) *orientation* output timing per Mtx sample @ 460k8 bps is:

(3*4 + 7) bytes * 10 = 190 bits

90 bits/460800 bits/s = 0.41 ms transmission time

Total time between physical event and receipt of complete data message:

6.43 ms + 0.41 ms = *6.84 ms* (worst case)

The *calibrated* output(9 floats) timing per Mtx sample @ 460k8 bps is:

(9*4 + 7)bytes * 10 = 430 bits

430 bits/460800 bits/s = 0.93 ms transmission time.

Total time between physical event and receipt of complete data message:

1.08ms + 0.93 ms = *2.01 ms* (worst case)

5.5 The Technique Correction Utility

In this section, I will provide concise details of how my C++ application is implemented. I will begin by describing the configuration and proceed to the main algorithm from there.

5.5.1 Configuration

Firstly, it was necessary to include the "IMTObj.h" header file, in order to import the functions supported by the MT Object. I also included "IMTObj_i.c", which handles the GUIDs of the MT Object. Next, I created a global pointer to the MTObj COM interface, as discussed previously in section 5.2.1. I have also specified the output format, to give me the orientation data in Euler angles. As I required timestamps for all the retrieved data samples, I enabled timestamped output, by setting the parameter to true(1). This done, I specified the reset type that I needed(0 for heading reset). The heading reset was done for reasons explained in section 4.5. Next, I created and initialised two arrays of type *float*. The first array was needed to hold the orientation data from both Mtxs', for one data sample, and the second array was used to hold calibrated data from both Mtxs'(also for one sample). The data from the two buffers previously mentioned , would be assigned to these two arrays before both buffers get cleared. I would then use these arrays to start filling the data samples into two larger 2D arrays. More details on this to follow.

All of these details from above can be seen in listing 5.2.

```
IMotionTracker* pMT;
```

```
short g_nMode = MT_LOGEULER;
short nTimeStamp = 1;
short nResetType = 0;
float fClubOrientationData[7] = {0.0};
float fClubCalibratedData[21] = {0.0};
```

Listing 5.2: *Configuration*

The next issue in the configuration process is to initialise the COM library. Listing 5.3 shows this.

```
printf("Initialize COM library.....................");
        if (CoInitialize(NULL) != S_OK)
                printf("Failed to initialize COM library");
        else
                printf("done\n\n");
```

Listing 5.3: *Initialising the COM library*

This enables the software component model discussed in section 5.2. The COM library is also uninitialised at program termination, and this is facilitated by the *CoUninitialize()* method.

The next stage in the configuration process is to create an instance of the MT Object and set all the necessary object parameters. This method is called after the COM library has been initialised. Listing 5.4 shows this method.

```
void SetupFilter()
```

```
{       // Set MTObj COM object options
        short bLogCalibratedData = TRUE;
        // Set MTObj COM object variables
        float fGain = 1.0;
        short nCorInterval = 1;
        float fRho = 1.0;
        short nPortNumber = 5;    //using com port 5 , spp over
    bluetooth link
        // Create instance of MTObj COM object
        printf("Create instance of MotionTracker object........");
        HRESULT hRes = CoCreateInstance(CLSID_MotionTracker, NULL,
    CLSCTX_SERVER, IID_IMotionTracker, (void**) &pMT);
        if (FAILED(hRes))
        {
                printf("Error %x in CoCreateInstance for MT object!"
    ,hRes);
                return;
        }
        else
                printf("done\n\n");
        printf("Setting filter parameters.....................");
        pMT->XM_SetCalibratedOutput(bLogCalibratedData);
        pMT->XM_SetTimeStampOutput(nTimeStamp);
```

```
        // Set Gain, Correction interval and Rho
        pMT->MT_SetFilterSettings(fGain,nCorInterval,fRho);
        pMT->XM_SetOutputMode(g_nMode);
        // Set COM port number where MotionTracker is attached
        pMT->XM_SetCOMPort(nPortNumber);
        printf("done\n\n");
}
```

Listing 5.4: *SetupFilter() creates an instance of the MT Object.*

In this method, I firstly set up the MT Object COM object variables. *fGain*, *nCorInterval* and *fRho* set the parameters necessary for the sensor fusion algorithm of the Mtxs'. These settings can also be tweaked. *fGain* is the filter gain which is the cross-over frequency of the sensor fusion algorithm in Hertz. The valid range for these values is greater than 0.01 and less than 50. By setting it equal to one, I am specifying that the frequency components of the calculated orientation vector exceeding 1Hz will be determined by the rate of turn sensors and the components below 1Hz will be determinded by the accelerometers and magnometers. *nCorInterval* is the correction interval and *fRho* is the weighting factor, which specifies how much sensor data should be weighted relative to the accelerometer data. I have specified a value of one for the *fRho*, which means that the magnometer data and the accelerometer data are considered equal. The next variable represents the COM port number on the laptop and I have fixed this to COM port number five. Then the MT Object instance is created by the *CoCreateInstance()* method, which appears next in listing 5.4. The

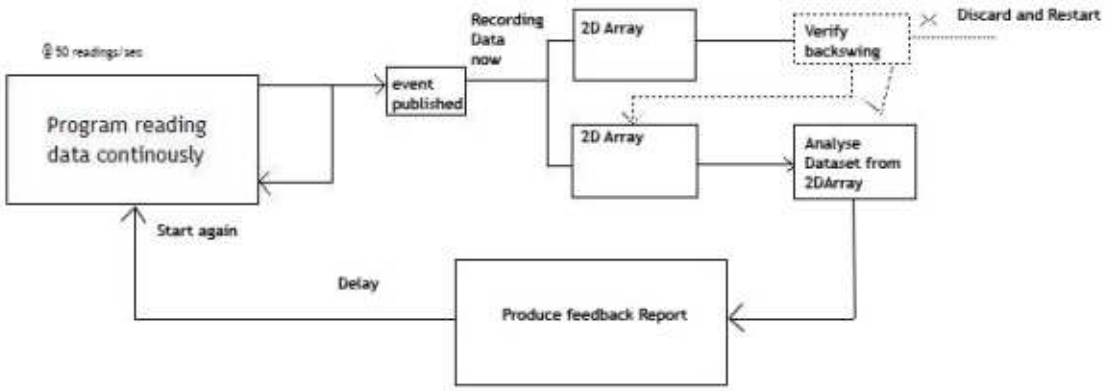

Figure 5.5: *The control flow of AnalyseMySwing*

next two methods in line *XM_SetCalibratedOutput()* and *XM_SetTimeStampOutput()* enable calibrated output and timestamped output, respectively. Note that there is a global pointer being used to access the MT Object interface here, as specified in section 5.2.1. *MT_SetFilterSettings()* applies the filter settings, *XM_SetOutputMode()* enables me to specify euler angles as the orientation output mode and *XM_SetCOMPort()* sets the COM port number where the XM is connected. After all the configuration is complete, the method *XM_StartProcess()* kick starts the processing of the motion trackers(Mtxs') by the MT Object.

5.5.2 AnalyseMySwing()

This section will deal with the main algorithm of the Sensor-Augmented club. The algorithm is named *AnalyseMySwing* and its control flow is shown in figure 5.5. Please refer to the CD attached, for the source code. The general aim of the main algorithm is

to recognised the action gesture of a golf swing. Upon recognising a correct backswing, the user receives feedback via the Technique Correction Utility. The orientation values of the clubface and the accelerometer readings are used in the analysis.

For analytical purposes, I have used two multi-script arrays, in which I save the sensor readings. There are also a number of booleans being used to indicate the success or failure of an event. Listing 5.5 shows the 2D arrays and some important booleans.

```
float swingArray[25][5] = {0.0};
float fullSwingArray[80][6] = {0.0};


bool swingHasStarted = false;
bool swingHasEnded = false;


bool fullSwingHasStarted = false;
bool fullSwingHasEnded = false;
bool goodShoulderTurn = false;
bool fullSwingReady = false;
bool goodTeeShot = false;
bool goodFollowThrough = false;
```

Listing 5.5: *Arrays and Booleans used.*

From Listing 5.5 you can see two arrays. *swingArray* stores twenty-five sample readings at a time. As the sensor is set to retrieve fifty samples per second, *swingArray* stores enough values to record half a second of data readings. Likewise, *fullSwingArray* will

record values for the duration of 1.6 secs.

Data will be placed into both of these arrays simultaneously, depending on the triggering of an event(explained later). *swingArray* is concerned with recording and analysing the data for a backswing, as can be seen from figure 5.5. Twenty-five data samples allow a complete backswing to be recorded and this also provides real-time recognition of the backswing gesture, as the computations on this small array are quite fast. The first two booleans are related to *swingArray* and are used to specify if the backswing has started or ended. The second bunch of booleans are all concerned with the second array(*fullSwingArray*), which as the name suggests, records data for the entire swing. Timestamped output is enabled, and index[0] in both arrays will record a timestamp float. There is also a counter used to iterate through the process of filling both arrays.

On running the application, an initialisation period of five seconds is facilitated with a sleep command. This allows the user some time in order to assume, the positioning for the swing, before the club orientation is reset(as mentioned in section 4.5). Once this timer expires, a heading reset is done with the *XM_ResetOrientation()* function.

The next step in the process is to place the Mtxs in measurement state, allowing them to start filling up and clearing their respective buffers with data samples. This process will run continuously in a loop as previously mentioned in section 5.3. next, I have specified a software event and when it is triggered, it starts the recording process of both arrays. Listing 5.6 begins by placing the Mtxs in a continuous measurement state, the method *GetData*, retrieves an orientation sample and *GetCalibrationData()* retrieves a calibrated reading sample.

```
while (bKeepRunning == TRUE && !_kbhit()){


bKeepRunning = GetData(fClubOrientationData);
bKeepRunning = GetCalibrationData(fClubCalibratedData);


//A trigger to start recording data
if(fClubCalibratedData[2] <= -10  && fClubCalibratedData[3] > 7 &&
   fClubOrientationData[2] <= 0.0
&& fClubOrientationData[2] >= -15 && swingHasStarted == false){
        swingHasStarted = true;
        fullSwingHasStarted = true;


}
```

Listing 5.6: *Data retrieval and event trigger.*

When the first sample comes along, which satisfies the conditions listed in the trigger, the event of both the backswing and complete swing has started. The trigger specifies an acceleration threshold in the -Y axis(direction of a backswing) and also the orientation threshold of the clubface, which when combined means that a possible swing gesture has started. Most of the parameter values for all the atomic thresholding in the *AnalyseMySwing()* method were given to me by during the swing analysis.

I will now discuss details on the small array, which can been triggered to record

the backswing. Using another control loop, once the event has been triggered, the next twenty-five data samples will be recorded into the *swingArray*. These data smples include the *Roll* and *Yaw*, which aid me to check the clubface orientation after half a second into the backswing. This part of the swing is the slowest and tests show that a correct backswing will always adhere to a small range of values, specified during the analysis of the swing. Once the array is full of data samples, I check the last data sample to see if it falls within the correct threshold range. If it does, a backswing is detected and the program proceeds from there, otherwise a possible notification of an error will be outputted to the user.

I mentioned earlier that both arrays, begin recording data together. A successful detection of a backswing, will mean a full analysis of the second array, otherwise the second array is discarded and a new valid backswing will be needed in the next swing. This second array records the orientation values of the club and also for the shoulder Mtx. A good shoulder turn will be recognised when its orientation values show that it has rotated from its original position through to the midpoint of the users stance. There is also a statement which records the clubs top acceleration in the Y axis(downswing). A method to compute the club speed in Km/ph is executed later. Both shoulder orientation and club speed values are recorded during the complete swing. Listing 5.7 shows how this is implemented.

```
if(fClubOrientationData[6] <= -100){
 goodShoulderTurn = true;
}
```

```
if(fClubCalibratedData[2] > topAccel){
topAccel = fClubCalibratedData[2];
}
```

Listing 5.7: *Shoulder and club speed check.*

When the full swing has ended and its data has been recorded, it is ready for processing. Firstly, the top speed of the club during the downswing will be produced to the user. Secondly, the orientation of the club, at the teeshot will be checked. For this, the *Roll* and the *Yaw* are the significant values. The club orientation for the teeshot will be checked from roughly one second into the swing onwards. This was done as the swing analysis showed that the crossover point always occurs in this range. Tests show that after the backswing, both will crossover from being negative values, to positive values. When the cross over point of these readings occur within six timestamps of each other, the *Roll* and *Yaw* have come back to the reset position successfully, indicating that good contact will be made with the ball. If the readings' cross-over point occurs six or more timestamps apart, then the execution has been poor and the algorithm will deduce, if the teeshot has been good or dragged wide. Checks have been put into place to make sure that a backswing must be detected and verified, in order to continue analysis on the teeshot and follow-through of the complete swing. This is implemented using the booleans, illustrated in listing 5.5.

The next form of feedback produced in the report will be the outcome of the rotating shoulder. If this has been deemed to cross over the midpoint of the users stance, it displays that the shoulder movement for the swing has been good. Other wise it will

output the error and a performance tip.

Lastly, the detection for a good follow-through is implemented over the last half a second of the swing. If a teeshot has been recognised, a follow-through will be analysed. If the orientation in this period adheres to a small range of atomic threshold values, then a correct follow-through will be recognised. These values were noted during the swing analysis and denote the clubs possible orientation within the last half a second of the swing. The intention here is to make sure that the user does follow-through with his swing, which compliments the correct technique.

5.5.3 Program Termination

Upon pressing "q" at any time, the program will quit and the *XM_StopProcess()* will stop processing by the motion tracker object(this will free up computing resources). Finally the motion tracker object is released and the COM library is uninitialised The released MT Object will unload itself from memory. The above can be seen in listing 5.8.

```
// Release and clean up MotionTracker object
        printf("Release MotionTracker object...");
        if (pMT != NULL)
        {       pMT->Release();
                pMT = NULL;
                printf("done\n\n");
        }
        // Uninitialize COM library
```

```
        printf("Uninitialize COM library...");
        CoUninitialize();
        printf("done\n\n");
```

Listing 5.8: *Releasing the MT Object and uninitialising the COM library.*

5.6 Summary

In this chapter I have introduced the COM middleware necessary for communication between the MT Object and the Xbus Master. I have also discussed the reasons for choosing event based communication. I have given details on the communication timing in relation to Mtx data acquisition and retrieval. I have presented details on the "Technique Correction Utility", with subsections dedicated to the configuration, main algorithm and program termination respectively. The next section will take a look at evaluating the project and some user scenarios will also be explored.

Chapter 6

Evaluation

In this chapter I will evaluate the project in terms of objectives achieved, and some user tests.

6.1 Objectives achieved

In section 1.5, I outlined the objectives of my dissertation. The five objectives were:

1. To record and store all of the orientation and calibrated dataset produced by a swing.

2. To analyse the dataset with respect to an algorithm, which determines the level of accuracy achieved, in line with the dataset of a perfect swing.

3. To recognise and analyse the backswing, teeshot and follow-through of a swing.

4. To produce a report for the user indicating where his/her technique is failing and to provide feedback on how to improve the technique based on errors made.

5. To Analyse some of the upper body movement using an additional sensor during the swing, to aid the development of a correct technique.

I believe that the objectives have been achieved in this project. The Technique Correction Utility successfully records all of the orientation and calibrated data for the swing. This application parses and analyses the streams of data, and interprets the information as it is trained to do. The application has the ability to recognise and verify the different components of a swing, as set out in objective 3. It also has the ability to provide basic feedback to the user, in terms of technique correction. More details on this will follow in section 6.2. The application also successfully provides additional feedback on shoulder movement during a swing.

6.2 User Testing

Testing was carried out back at the driving range. Three users participated in carrying out test swings using the Xbus Master, the shoulder Mtx and the Sensor-Augmented club. Testing was carried out on the driving mats, where hitting the tee was deemed sufficient. We avoided using a golf ball in order to avoid any damages to the Xsens equipment.

6.2.1 Senario A

Scenario A shows details of three test swings carried out by my former tutor. The *Roll* and the *Yaw* of the Sensor-Augmented club have been outputted, to elaborate further on section 5.5.2. Figure 6.1 shows a printscreen of the application after three test

swings were completed. You can see from figure 6.1, that the tutor achieved perfect results from the Technique Correction Utility, due to his accurate technique.

6.2.2 Senario B

Figure 6.2 shows a printscreen after scenario B was carried out. This user was an amateur and his best swing was produced in his first attempt. From figure 6.2, you can see that on his first swing, his backwing was successfully detected and hence the report was produced. The application provides feedback on club speed, club orientation at teeshot, shoulder movement and the follow-through. Each of these were successful on his first attempt. During his second attempt, the user let his club slip, and the application recognised an error in his backswing. On the his third attempt, he was given feedback to correct his shoulder movements and also the follow-through.

6.2.3 Senario C

Figure 6.3 shows a printscreen after scenario C was carried out. The user involved in senario C was a semi-pro who was practising at the driving range. His three attempts can be seen in figure 6.3. In his first attempt, his swing was interpreted as incomplete as the follow-through was not good. In his second swing, his follow-through was good but his shoulder movement was inadequate. His third effort was technically sound and the Technique Correction Utility verified this.

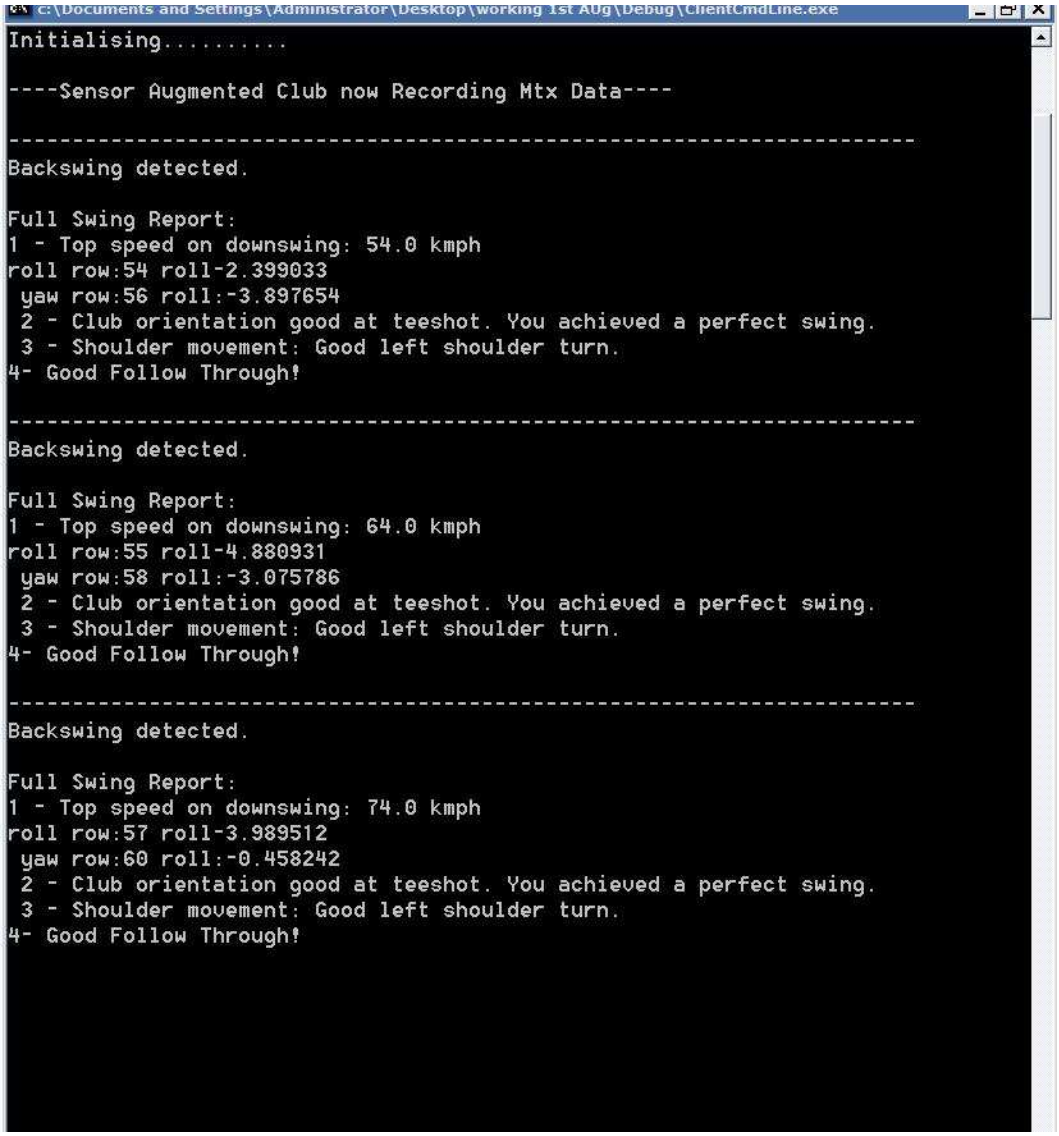

Figure 6.1: *A printscreen of scenario A*

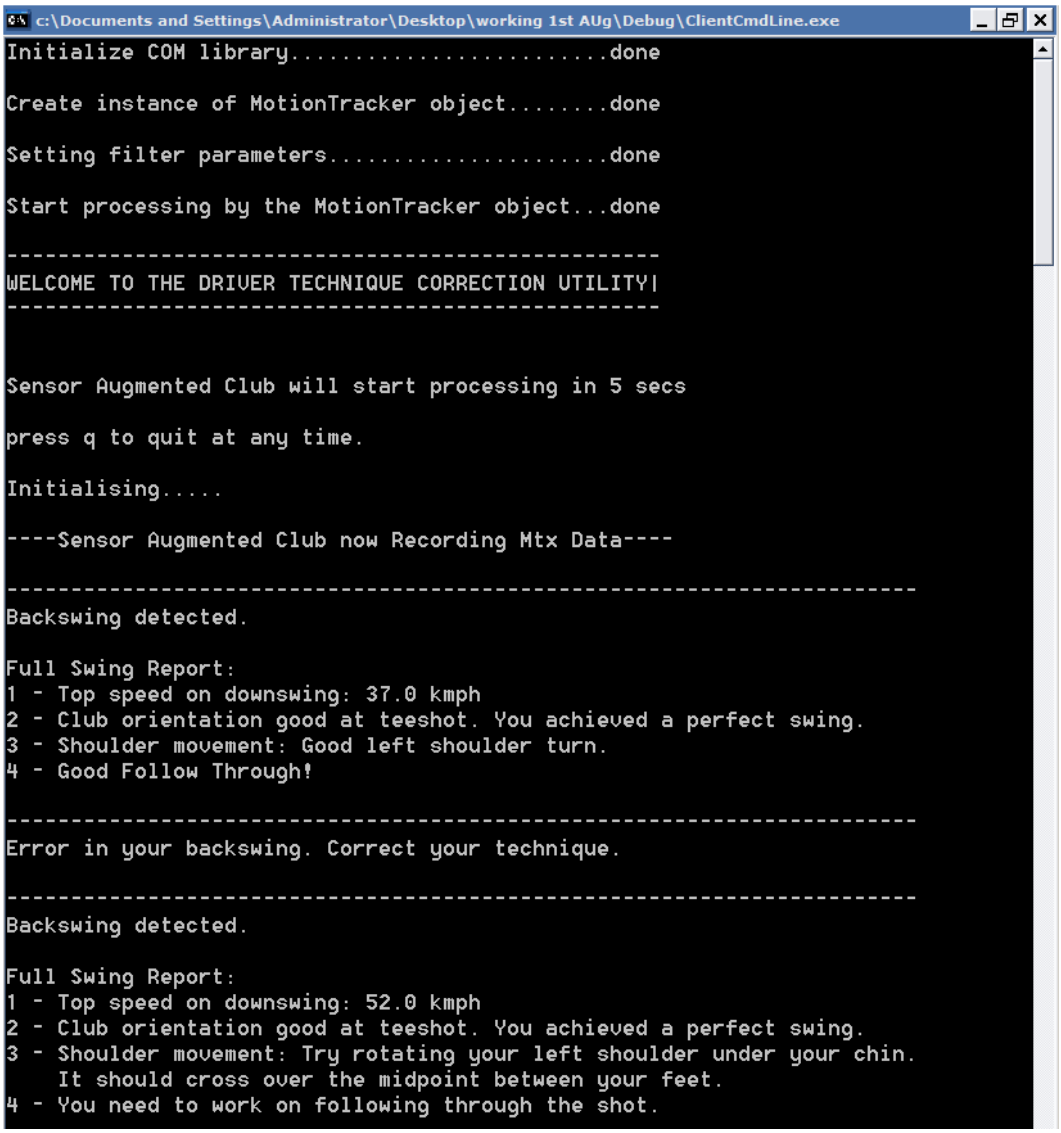

Figure 6.2: *A printscreen of scenario B*

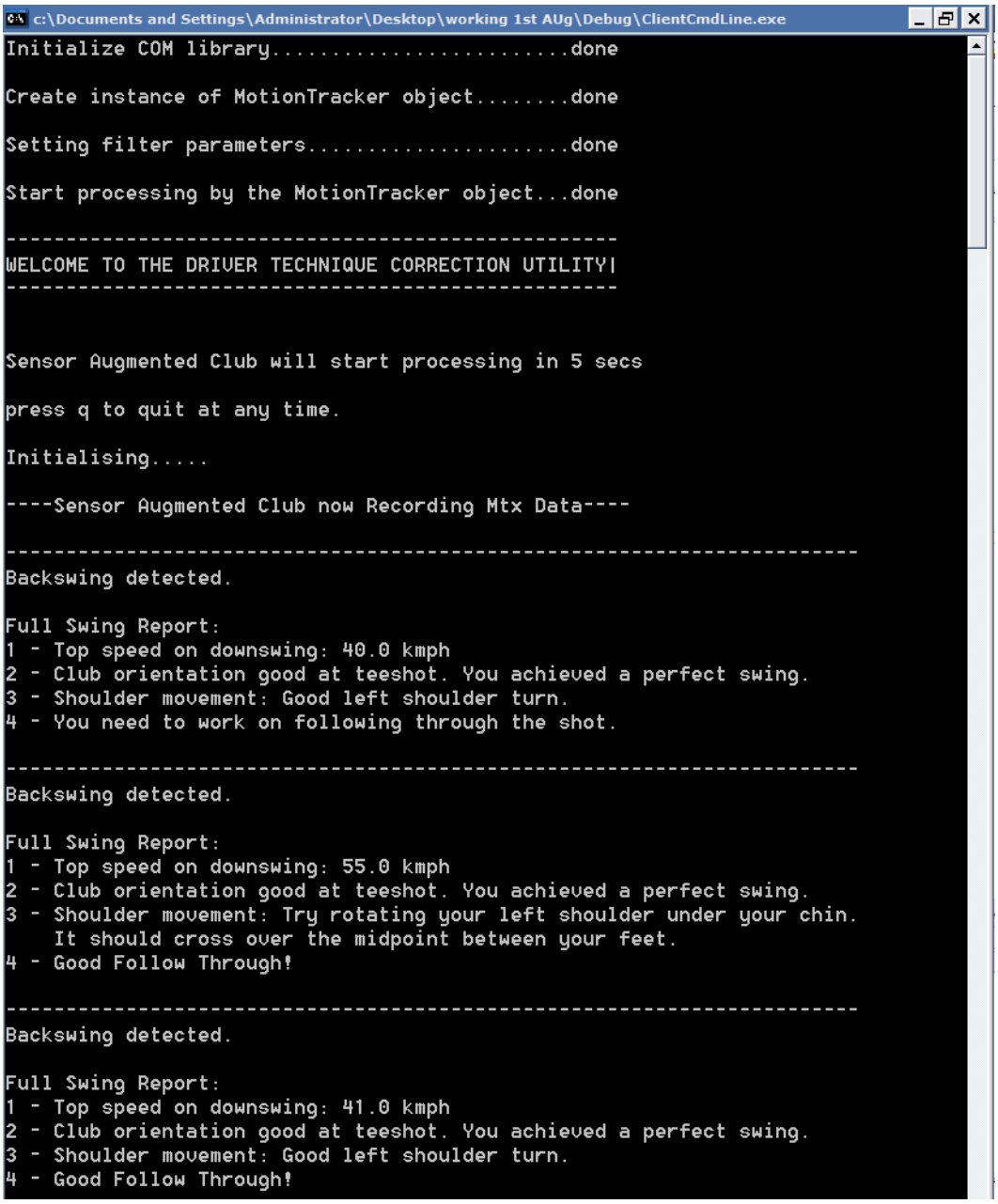

Figure 6.3: *A printscreen of scenario C*

6.2.4 Brief questionnaire

After the user testing was carried out. I requested some feeback and five people provided their opinions on the following questions. Three of these users were involved in the scenarios. Below are the questions.

Do you feel golf beginners can benefit from this sensor-augmented artefact?(Y/N)

To what extend do you believe that sports training can be aided by sensor-based applications?(Strong belief/ Unsure/ Don't Know)

Overall impression of the application?(Good/OK/Bad)

Could you provide any ideas on improving this application?

Four out five users felt this artefact could become useful and may be present in the driving ranges of the future.

Three out of five users felt that sports training can be enhanced by the use of sensor based application. Two users were unsure.

Three users said that the overall impression of the application was good. Two users said it was "OK".

Amongst the feedback on improving the application was the idea of having a better "GUI""[1] for the user.

[1]Graphical User Interface

6.2.5 Summary

In this chapter I evaluated the objectives of the dissertation. I conducted three user scenarios and described their findings. I have also discussed the feedback given to me via the brief questionnaire.

Chapter 7

Conclusions

This chapter looks at the overall dissertation and possible future work. From my review of the state of the art, I became quite aware and fascinated at the prospect of implementing a sensor based artefact. Some of the work in chapter 2 has inspired the development of the Sensor-Augmented club.

The vision of Ubiquitous Computing includes context-aware small mobile devices. This vision also includes artefacts which can be an everyday object, such as the Media Cup in section 2.4.3 or such as the Sensor-Augmented club. This is an ordinary club, but sensory technologies have allowed it to extend its standard functions.

The purpose of the Sensor-Augmented club is to help with driver training, by providing training tips to the user. The application is just a prototype at the moment but I have demonstrated that sensor-augmented artefacts can help us do something, and that they are likely to become ubiquitous in our environments over the next few years.

In chapter one of this thesis I have given you an overview of golf, the club utilised in the project and my objectives. In the second chapter I have described a diverse range of applications which relate the latest work done using sensors. In the second chapter, I have shown that a broad range of applications can be developed using sensor technologies. With the use of sensors, applications and ordinary artefacts can appear more smart, facilitating a new means of human computer interaction.

In chapter three I have given a system design overview and introduced you to the hardware deployed in my project. Chapter four looked at the basic components and technique involved in a golf swing. I also hightlighted the data acquisition process via the experiments. In chapter five I gave technical information on the underlying communication methods deployed between the sensor and the application. I provided details on the application's configuration, main algorithm and program termination, with the use of some listings. I chapter six I have given the project an evaluation with the aid of some test scenarios.

7.1 Future Work

This section outlines some future work initiatives which could be applied to the Sensor-Augmented club.

7.1.1 Interactive GUI

Given the responses I got from the brief questionnaire, I feel that this prototype application could become a more popular training program, by providing a better interface

and possible extending its functionalities. A windows MFC type application could facilitate this. Also, creating charts and statistics could be facilitated by MS Excel or Matlab using the COM object. The charts and statistics could be used further by coaches , or used by golf specific software developers, for further analysis.

7.1.2 Game Interface

The application as it stands, has the ability to recognise a backswing in real-time. It also recognises the other pivotal points of a swing with little lag. Compared to commercial products such as "Electric Spin"[5][1], the performance of the Sensor-Augmented club would be 3-4 seconds quicker. The Sensor-Augmented club recognises a backswing, teeshot and follow-through. Upon recognition, it could send commands to a games console via the bluetooth protocol or by wireless radio frequency.

7.1.3 An artefact for Virtual Reality Golf.

It maybe be possible to build an application where the movements of a 3D golf club model could be updated in real-time. This project would be similar to the smart sword's visualisation application from chapter two. This opens up a new range of virtual reality golf applications. This would extend the functionality of the sensor-augmented club and give birth to a new virtual reality experience.

[1]Electric Spin is a game interface for PC's and the Play Station 2. It consists of the mat and a ball. Using optical sensors it predicts the ball trajectory in the game.

Bibliography

[1] *Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model*, 2002.

[2] D. Becker. Sensie: A real-time recognition, feedback and training system for t'ai chi gestures. Technical report, M.I.T Media lab Perceptual Computing Group, May 1997.

[3] G. Chambers, S. Venkatesh, G. West, and H. Bui. Hierarchical recognition of intentional human gestures for sports video annotation. In *Pattern Recognition 2002. Proceedings. 16th International Conference on*, volume 2, pages 1082–1085, 2002.

[4] G. S. Chambers, S. Venkatesh, and G. A. W. West. Automatic labeling of sports video using umpire gesture recognition. In *SSPR/SPR*, pages 859–867, 2004.

[5] E. S. Corporation. Electric spin. http://www.electricspin.com, 2006.

[6] J. M. Davies. The aerodynamics of golf balls. In *Journal of Applied Physics*, volume 20, September 1949.

[7] M. Fay, M. Woulfe, S. Tsvetkov, and M. Haahr. Smart couch. http://www.dsg.cs.tcd.ie/node/171, 2003.

[8] A. Gellersen and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artefacts, 2002.

[9] T. Jorgensen. *The Physics of Golf*. The American Institute of Physics, March 1999.

[10] K. Moon, Y. Lee, Y. Son, and C. Kim. Universal home network middleware guaranteeing seamless interoperability among the heterogeneous home network middleware. In *IEEE Transaction on Consumer Electronics*, volume 49, pages 546–553, 2003.

[11] R. Penner. Progress in physics. Technical report, Malaspina University-College, British Columbia, USA, 2003.

[12] H. L. Power and J. D. Iverson. Magnus effect on spinning bodies of revolution. In *AIAA Journal*, volume 11, pages 417–418, April 1973.

[13] M. Roman, F. Kon, and R. H. Campbell. Reflective middleware: from your desk to your hand. Illinois, USA.

[14] L. Rudolph. Project oxygen: Pervasive, human-centric computing - an initial experience. In *CAiSE '01: Proceedings of the 13th International Conference*

on Advanced Information Systems Engineering, pages 1–12, London, UK, 2001. Springer-Verlag.

[15] X. Technologies. The xbus master and mtx. http://xsens.com, 2006.

[16] R. Weber-Collins, S. Tsvetkov, and M. Haahr. Smart sword. http://www.dsg.cs.tcd.ie/node/172, 2003.

[17] M. Weiser. The computer for the 21st century. *Scientific American*, 1991.

[18] M. Yang and N. Ahuja. Extraction and classification of visual motion patterns for hand gesture recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 892–897, Santa Barbara, California, 1998.