An Inertial Motion Capture Framework for Constructing Body Sensor Networks

by

Tudor Pascu

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy
at the University of Sussex

School of Engineering and Informatics
Department of Informatics
University of Sussex
Brighton
BN1 9QT

February 2015

Declaration

The work described in this thesis, carried out in the School of Engineering and Informatics, is that of the author and has not been submitted in any form for any other degree at this or any other university.

Signed ____________________

Tudor Pascu

School of Engineering and Informatics
Department of Informatics
University of Sussex
Brighton
BN1 9QT

Acknowledgements

First and foremost, I would like to thank my supervisors Dr. Martin White and Dr. Paul Newbury and everyone in the School of Engineering and Informatics who have provided support and guidance in the completion of this degree.

Additionally, I would like to thank Cristi Ureche, Alexandru Cotut and my father, Liviu Pascu, for providing me with their expertise and facilities for developing the hardware behind the motion capture system presented in Chapter 4.

Many thanks go out to all family and friends for their words of encouragement and moral support throughout this process.

University of Sussex

Tudor Pascu

Submitted for the degree of Doctor of Philosophy

An Inertial Motion Capture Framework for
Constructing Body Sensor Networks

Abstract

Motion capture is the process of measuring and subsequently reconstructing the movement of an animated object or being in virtual space. Virtual reconstructions of human motion play an important role in numerous application areas such as animation, medical science, ergonomics, etc. While optical motion capture systems are the industry standard, inertial body sensor networks are becoming viable alternatives due to portability, practicality and cost. This thesis presents an innovative inertial motion capture framework for constructing body sensor networks through software environments, smartphones and web technologies.

The first component of the framework is a unique inertial motion capture software environment aimed at providing an improved experimentation environment, accompanied by programming scaffolding and a driver development kit, for users interested in studying or engineering body sensor networks. The software environment provides a bespoke 3D engine for kinematic motion visualisations and a set of tools for hardware integration. The software environment is used to develop the hardware behind a prototype motion capture suit focused on low-power consumption and hardware-centricity. Additional inertial measurement units, which are available commercially, are also integrated to demonstrate the functionality the software environment while providing the framework with additional sources for motion data.

The smartphone is the most ubiquitous computing technology and its worldwide uptake has prompted many advances in wearable inertial sensing technologies. Smartphones contain gyroscopes, accelerometers and magnetometers, a combination of sensors that is commonly found in inertial measurement units. This thesis presents a mobile application that investigates whether the smartphone is capable of inertial motion capture by constructing a novel omnidirectional body sensor network.

This thesis proposes a novel use for web technologies through the development of the Motion Cloud, a repository and gateway for inertial data. Web technologies have the potential to replace motion capture file formats with online repositories and to set a new standard for how motion data is stored. From a single inertial measurement unit to a more complex body sensor network, the proposed architecture is extendable and facilitates the integration of any inertial hardware configuration. The Motion Cloud's data can be accessed through an application-programming interface or through a web portal that provides users with the functionality for visualising and exporting the motion data.

List of Publications

Journals

Tudor Pascu, Martin White, Natalia Beloff, Zeeshan Patoli, Leon Barker, "Ambient Health Monitoring: The Smartphone as a Body Sensor Network Component", *The Journal of Innovation Impact*, 6 (1), pp. 62-65, 2013, ISSN: 2051-6002.

Claire Nee, Martin White, Kirk Woolford, **Tudor Pascu**, Leon Barker, Lucy Wainwright, "New methods for examining expertise in burglars in natural and simulated environments: preliminary findings", *Psychology, Crime & Law*, December 2014, ISSN 1068-316X.

Conference Proceedings

Sasithorn Rattanarungrot, Martin White, Zeeshan Patoli, **Tudor Pascu**, "The Application of Augmented Reality for Reanimating Cultural Heritage", *6th International Conference on Virtual, Augmented and Mixed Reality, Held as Part of HCI International 2014*, pp 85-95, Crete, Greece, June 2014, ISBN: 978-3-319-07463-4.

Leon Barker, Martin White, Mairead Curran, Zeeshan Patoli, Benjamin Huggins, **Tudor Pascu**, Natalia Beloff, "Taxonomy for Internet of Things: Tools for Monitoring Personal Effects", *Proceedings of 4th International Conference on Pervasive and Embedded Computing and Communication Systems*, Lisbon, Portugal, January 2014, ISBN: 978-989-758-0000-0.

Martin White, Zeeshan Patoli, **Tudor Pascu**, "Knowledge Networking through Social Media for Digital Heritage Resources", *Proceedings of the International Congress on Digital Heritage*, Marseille, France, November 2013, ISBN: 978-1-4799-3169-9/13.

Tudor Pascu, Zeeshan Patoli, Martin White, "Motion Capture and Activity Tracking Using Smartphone-Driven Online Body Sensor Networks", *Proceedings of 3rd International Conference on Innovative Computing Technology (INTECH)*, London, United Kingdom, August 2013, ISBN: 978-1-4799-0047-3 (best paper award).

Tudor Pascu, Zeeshan Patoli, Leon Barker, Natalia Beloff, Martin White, "Ambient Health Monitoring: The Smartphone as a Body Sensor Network Component", *Proceedings of Innovation in Medicine and Healthcare (INMED)*, pp. 62-65, Piraeus, Greece, July 2013, ISBN: 978-0-9561516-3-6.

Tudor Pascu, Zeeshan Patoli, Martin White, "Improving Anchor Selection for Inertial Motion Capture Systems Through Weight Distribution Calculations", *Proceedings of 13th International Conference on Computer Graphics and Imaging (CGIM)*, Innsbruck, Austria, February 2013, ISBN: 978-0-88986-954-7.

Tudor Pascu, Zeeshan Patoli, Martin White, "Unifying Software and Hardware-Centric Inertial Measurement Units in Body Sensor Networks", *Proceedings of 13th International Conference on Computer Graphics and Imaging (CGIM)*, Innsbruck, Austria, February 2013, ISBN: 978-0-88986-954-7.

Workshops, Invited Talks and Poster Presentations

Claire Nee, Martin White, Kirk Woolford, **Tudor Pascu**, Leon Barker, "Examining Expertise In Residential Burglars: The Results Of A Pilot Study Using Innovative Technology", *68th Annual Meeting of the American Society of Criminology (ASC)*, Chicago IL, November 2012.

Leon Barker, Zeeshan Patoli, **Tudor Pascu**, Martin White, "Using Web3D to Integrate Motion Capture Data With 3D Visualization - Facilitating Historic Re-enactments Through the Web Browser", presented at workshop *Computer Applications in Archaeology Conference (CAAUK)*, Birmingham, United Kingdom, April 2011.

Martin White, Claire Nee, Zeeshan Patoli, **Tudor Pascu**, "Virtual Burglary Simulation", presented at workshop *Virtual Emergencies: Simulation Technology for Emergency Planning and Response*, Royal United Services Institute, London, United Kingdom, April 2011.

Table of Contents

Table of Figures

Tables

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
4D	Four-Dimensional
AHRC	Arts and Humanities Research Council
AHRS	Attitude Heading Reference System
AMC	Acclaim Motion Capture
API	Application-Programming Interface
ARW	Angular Random Walk
ASF	Acclaim Skeleton Format
BSN	Body Sensor Network
BVA	Biovision File Format
BVH	Biovision Hierarchy
BVHE	Biovision Hierarchy Extended
CRC	Cyclic Redundancy Check
CSS	Cascading Style Sheet
CSV	Comma-Separated Values
DDK	Driver Development Kit
DLL	Dynamic-Link Library
DOF	Degrees of Freedom
GFINS	Gyro-Free Inertial Navigation System
GPU	Graphics Processing Unit
HCI	Human Computer Interaction
IDC	Insulation-Displacement Connector
IMU	Inertial Measurement Unit
JSON	JavaScript Object Notation
LACOMA	Lowest-Anchor Centre of Mass Algorithm
LED	Light Emitting Diode
LSB	Least Significant Bit
MEMS	Micro-Electro-Mechanical Sensor
MCML	Motion Capture Markup Language
MIPP	Motion in Place Platform
MTDS	Motion Tracking Development System

MVC	Model View Controller
RAM	Random Access Memory
SD	Secure Digital
SDK	Software Development Kit
TSB	Technology Strategy Board
UML	Unified Modelling Language
VCP	Virtual Communication Port
WBAN	Wireless Body Area Network
WebGL	Web Graphics Library
XML	eXtensible Markup Language

CHAPTER ONE

1 Thesis Overview

1.1 Introduction

Motion capture is the process of measuring and subsequently reconstructing the movement of an object or being in virtual space. For the past decade, filmmakers and game developers have adopted motion capture technologies to animate computer-generated characters time-efficiently to achieve a superior level of realism. Although animation is the primary use, motion capture has become ubiquitous throughout the fields of biomechanics, ergonomics, medical science, sport science, aviation, etc. Motion capture is generally targeted at replicating and understanding human locomotion as the musculoskeletal properties of the body are highly complex.

At present, there are many technologies available commercially for tracking human movement that vary in performance, affordability and practicality. The most popular technologies are inertial, optical and mechanical. Each technology combines hardware and software to interpret motion as accurately as possible. For example, optical systems use computer vision algorithms to process video streams recorded by cameras. Multiple cameras can be used, often in conjunction with visual markers such as flashing light-emitting diodes or reflective points, to triangulate the position of joints and subsequently determine the orientation of body parts. As an alternative, inertial motion capture uses inertial measurement units (IMU) to capture individual gyrations of body parts. A full-body sensor network (BSN) will contain as much as twenty IMUs integrating gyroscope, accelerometer and magnetometer sensors. Angular readings are collected from each sensor and the result is displayed using a kinematic simulation model [1]. Kinematic models are skeletal rigs used to drive character topologies in virtual environments.

Accuracy is a measure of the difference between the real movement and its virtual reconstruction. In terms of accuracy, each motion capture medium presents specific qualities and limitations. For instance, optical motion capture limits the recording process to specific environments that present optimal lighting condition [2]. Additionally, any visual obstructions can cause occlusion problems. In contrast, inertial motion capture is unaffected by its recording environments, but suffers from hardware induced inaccuracies (e.g. sensor drift, calibration, dead reckoning, signal noise, etc.). Magnetic interference can also be a problem in factories or industrial environments [3] [4].

The earliest trace of inertial motion capture dates back to the beginning of the eighteenth century when Johan von Bohenenberger, a German professor of mathematics and astronomy, invented the first mechanical gyroscope. Leon Foucault, a French physicist, later took this concept and used it to measure the Earth's rotation. The invention consists of a spinning disc attached to gimbals that can rotate freely in all three axes. The centrifugal force of the spinning disc keeps its orientation fixed while the device is moved, thus measuring angular rotations. Throughout the nineteenth century, the aviation industry used this principle to mass-produce micro-electro-mechanical sensor (MEMS) gyroscopes that measure the yaw, pitch and roll of aircrafts during flight. Modern-day inertial motion capture has become ubiquitous in mobile computing technologies because of the miniaturization of sensor chips and the availability superior computational resources.

Inertial motion capture was chosen as the focal topic of this thesis because it proves challenging in many aspects. The process of recording and subsequently reconstructing motion is very rapid and complex. Data acquisition, angular conversions, computer-hardware intercommunications, data cleaning and kinematic deployment are processes that take place over thirty times per second without noticeable latencies. This thesis argues that the act of using inertial motion capture systems is a multifaceted sequence of procedures that lack standardisation and can therefore be improved. The average user does not present the knowhow to operate BSNs without studying a complicated instructions manual. There is no general-purpose platform for integrating, customizing and developing BSNs for experimental research. As a result, the main objective of this work is to develop a cross-platform framework for acquiring and processing data from commercial animation suits, mobile computing technologies and most hardware devices that output inertial data. The design and development is focused on enhancing the adaptability, modularity and overall usability of inertial BSNs by optimizing the motion capture workflow. The term workflow can be used to summarise the series of procedures required to operate an inertial motion capture suit at both software and hardware levels. Sensor hardware is complex in nature and does not follow the plug-and-play standard of other computer peripherals. The workflow for motion capture suits is often overcomplicated, unstandardized and requires an additional person to be present to operate the system. The ultimate goal of this thesis is to standardise the interaction between users inertial motion capture system by proposing an improved and more simplified workflow.

1.2 Problem Statement

This problem statement is the direct result of the literature survey of Chapter 2 and the critical analysis of principal industry contributors. Additionally, this problem statement also takes into consideration the former experience of the University of Sussex Interactive Systems Group in the field of inertial and exoskeleton motion capture (see Section 1.6). Let's consider a usability scenario that highlights the challenges faced in using a modern inertial motion capture.

Typical Inertial Motion Capture Scenario:

> A motion performer and a system operator use a motion capture suit to record full-body motion for the period of one hour. The purpose of the recording is to analyse human-environment interaction throughout urban households. The performer is dressed in the hardware with the help of the operator. This process takes approximately half an hour as nineteen IMUs and their interconnecting cables must be secured firmly on the body in a particular configuration (in accordance to the instruction manual). The suit is switched on and the network handshaking begins whereby the computer interrogates all sensor nodes individually. The user is asked to stand next to a reference object, which is used by the operator to adjust the onscreen kinematics to match the motion performer's bodily proportions. The motion performer faces north and the magnetometers are zeroed. Performing the T-pose, which involves standing straight with both arms extended laterally away from the body, compensates the postural difference. The suit begins recording if all these steps are performed successfully. While navigating the household environment, sensors may switch off due to loose or damaged connectors. Additionally, the chance of the suit losing wireless signal due to distance or environment obstructions is high. A single sensor disconnecting implies a complete system restart, which

involves repeating the aforementioned procedures. Once the recording session has completed successfully, the operator takes the data and applies a set of filters and data cleaning algorithms. The result is clean motion that can be used for scientific analysis and is stored using one of several file formats.

The above scenario exemplifies the lack of flexibility in using an inertial motion capture system. The procedures for initiating an animation suit are multifaceted and specific to every system. While most computer peripherals follow the plug-and-play standard, inertial BSN require a sequence of procedures, performed both by the motion actor and the recording session operator, to begin outputting motion. Furthermore, inertial motion capture systems are designed to work in one predefined way that is usually aimed at character animation. There is little room for researchers to customize the hardware or software configuration. The following problem areas are centred on: software environments, software standardisations, file formats, robustness and dead reckoning.

1.2.1 Software Environments

The first problem area is focused on motion capture software standardisations and support. Prior to developing any solutions that may be beneficial to the field of motion capture, a standardised simulation and development environment is required. Because every system is different, there is no standardisation between software environments. The lack of standardisation makes the development of solutions difficult. At present, there is no general-purpose software platform that can provide the functionality required for researchers to explore, develop and objectively evaluate motion capture technologies. Existing commercial and open-sourced software applications are predominantly designed for animation purposes and provide little room for experimental research. Inertial motion capture will benefit from a purpose-built software environment that outputs data in a format that is suitable for experimental research.

1.2.2 Standards and File Formats

At present, motion data can be manipulated using several unstandardized file formats [5] [6] [7] [8] [9], none of which are comprehensive. Most formats are limited to basic skeletal definitions and angular readings. Developers have created supplementary file formats that contain profiling data whereby profiling data scales the kinematics so that the skeletal rig matches the performer's body in terms of proportions. Those file formats could be combined to produce a more complete file format that could ultimately become the industry standard. Although there are software solutions for storing and organising inertial motion capture data in repositories, the act of transferring data between computers implies passing files between users. There is a need for web-based repositories for manipulating motion capture data through application-programming interface (API) calls.

1.2.3 Robustness

As with all wearable hardware, inertial systems are fragile in nature. Sensors are likely to disconnect if a cable is damaged, especially during recording sessions in cluttered environments. A sensor disconnecting will likely cause the BSN to stall. Restarting the system, which is both time-consuming and impractical, will generally solve this problem. For behavioural studies that examine psychological and physiological aspects of the human body, restarting the hardware in the midst of an experiment can be detrimental to the overall results. This behaviour is caused by BSNs being software-centric whereby all the data is extracted from the hardware is sent to a computer for processing. Raw data is larger in size, requiring BSNs to communicate larger messages between nodes and to the computer. Having a software-centric system also requires extensive handshaking procedures between the hardware and the computer. For that reason, the act of initiating an inertial motion capture system is a complicated procedure.

1.2.4 Dead Reckoning

While optical tracking systems triangulate joint displacements to determine the spatial positioning of the motion performer, inertial systems contain no inherent sensors for dead reckoning. Software can be used to compute rough estimations of horizontal displacement by applying planar collision detection and foot placement estimation [10] [11] [12] to kinematic models. This methodology is limited to simple gait on flat surfaces where the motion performer is taking clear steps. As an alternative, peripheral sensors can be used to compute dead reckoning using ultrasound emitters [13] or optical cameras [14]. However, this approach is costly and limits the recording process to confined environments.

1.3 Framework Overview

When pieced together, the developments presented in this thesis form one framework entitled Skeletrix. The title is derived from the words *skeletal* and *matrix*. Skeletrix starts with sensing motion through electronics, continues with reconstructing animations in virtual space and finishes with uploading motion into online storage. As shown in Figure 1-1, the framework explores solutions for every stage of the motion capture workflow. While the framework is aimed at research and development, its goal is to improve the overall usability, modularity and versatility of inertial systems.

The five major framework components are divided between three mediums: hardware, software and online. Each medium corresponds to a portion of the motion capture workflow. At the *hardware* level, the framework integrates Motion Tracking Development System (MTDS), a prototype motion capture suit, and evaluates other sensor technologies that are available commercially to form a baseline for benchmarking. At the *software* level, the framework proposes a software environment and a mobile application providing the functionality for gathering, processing and visualising motion data. At the *online* level, the framework presents the Motion Cloud, an online repository for storing, organising and visualising inertial data.

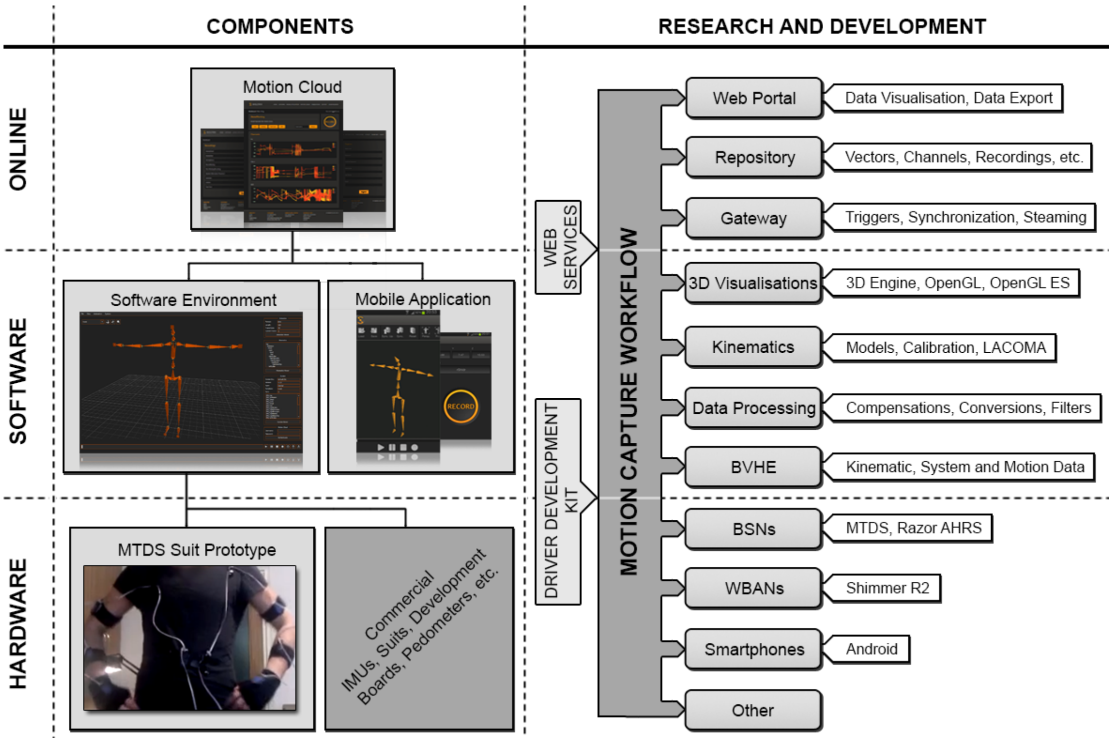


Figure 1-1: Overview of the Skeletrix framework. **Source: Pascu et al. [15] [16]**

1.4 Contributions to Knowledge

The contributions to knowledge presented in this thesis are in direct response to the problem areas previously identified and further research. In summary, this research presents a framework aimed at furthering the field of inertial motion capture.

As shown in Figure 1-2, the research approach presented in this thesis aims to examine and exploit existing technologies and principles as well as investigating new technologies that may be relevant to the field of inertial motion capture. The existing technologies researched are focused on understanding software applications and standards in order to present a unique set of contributions to knowledge (shown in blue) through the development of a bespoke and innovative software environment. To test that architecture and to investigate problems faced when developing new BSNs, this research examines and integrates existing sensor technologies as well as developing a new and innovative motion capture suit. The new technologies researched are mobile computing technologies (omnidirectional smartphone-driven BSNs) and web technologies (online repositories and gateways for motion data).

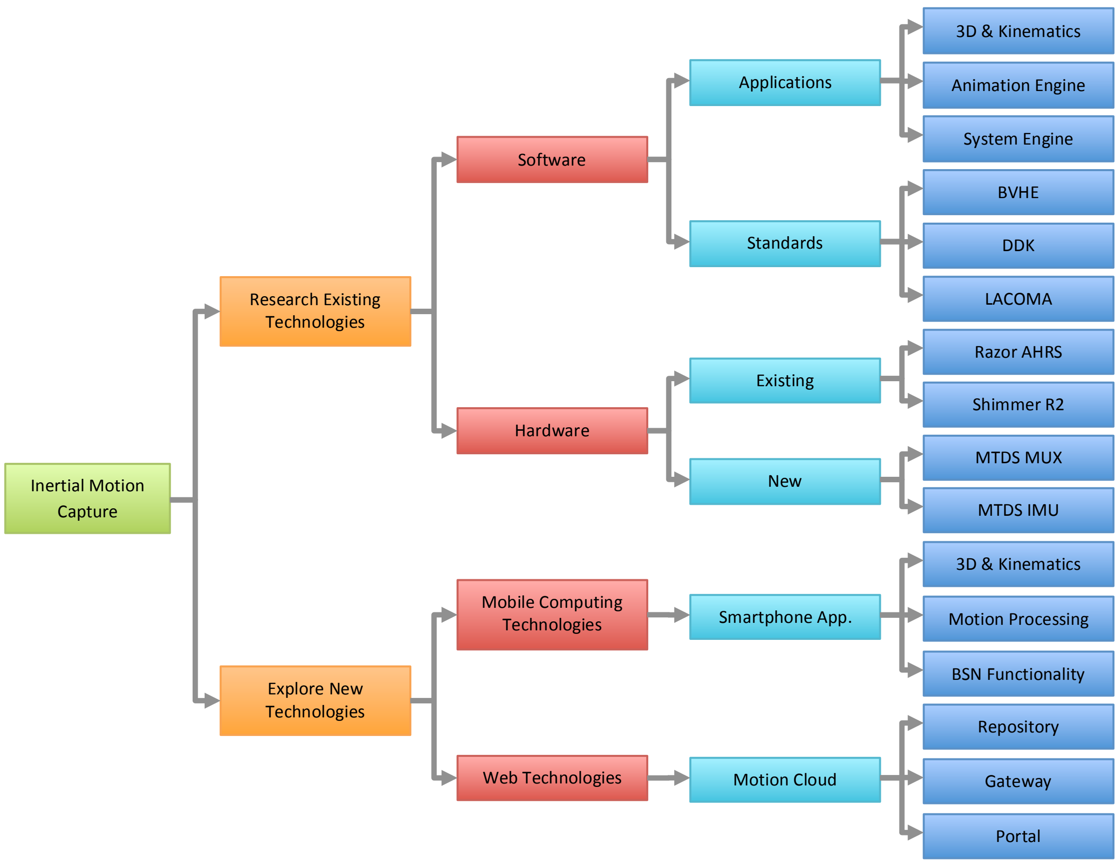


Figure 1-2: Overview of research approach and contributions to knowledge.

The framework developments can be divided into four research bodies, where each body corresponds to a thesis chapter, entailing: inertial motion capture software environments, constructing inertial BSN, sensing through mobile computing technologies and Motion Cloud: an repository and gateway for inertial motion data.

1.4.1 Inertial Motion Capture Software Environments

As previously published in Pascu et al. [17] [18], this work's first contribution to knowledge is a software environment and architectural scaffolding for researchers interested in studying, modifying or constructing inertial BSNs. The following application areas are also discussed to illustrate its versatility: hardware development, constructing heterogeneous BSNs and system benchmarking. The software environment provides the functionality for hardware integration, calibration, motion processing and three-dimensional (3D) visualisations. The development of this software environment was focused on three contributions to knowledge that are focused on standardising BSNs computer-hardware communications, motion capture file formats and kinematic dead reckoning.

The driver development kit (DDK) is a solution for hardware integration, which provides users with the means for developing driver modules that extract motion data from single IMUs or BSNs. The software environment can communicate with driver modules, which are self-contained dynamic-link libraries (DLL), to gather, process and visualise inertial motion data. Using the DDK allows users to develop BSNs that are more modular and customizable.

Biovision Hierarchy Extended (BVHE) is a proposed file format that contains skeletal definitions, motion data and system configurations in the same file. This unique format is presented as a solution for standardising the method of expressing the configuration of a BSN within a software environment. The advantage of BVHE over any existing format is practicality (simplifying the motion capture workflow) as users only have to load one file into the software environment. Given that kinematic hierarchy is driven by a system, which contains a system configuration, it is both logical and necessary to form a relationship between the two by establishing one singular and more complete file format.

This solution enhances the customizability of BSNs by tightening the relationship between software and hardware. Additionally, this extended file format is backwards compatible with existing software applications that support the traditional Biovision Hierarchy (BVH) file format.

The lowest-anchor centre of mass algorithm (LACOMA) Pascu et al. [18] is a dead reckoning algorithm for computing more accurately the correct point of support of the body during gait. It combines the concept of weight models with planar collision detection. This approach is aimed at improving anchor selection in kinematic models by using the body's musculoskeletal centre of weight to improve foot placement estimation. This solution improves accuracy, reduces the probability of anchor selection errors and is computationally inexpensive.

1.4.2 Constructing Inertial Body Sensor Networks

This work's second contribution to knowledge is a study focused on understanding how to improve inertial motion capture hardware through the development of the Motion Tracking Development Suit (MTDS) Pascu et al. [18] prototype, which is also used to demonstrate the functionality of the software environment. Two additional IMUs that are available commercially are also integrated to form a comparison. MTDS is a hardware-centric BSN developed by the author of this thesis and SC GPS Communications SRL, a Romanian hardware manufacturer. Because of recent advances in affordable micro-electro-mechanical sensor technologies, the suit was developed cost-effectively while focusing on low-power consumption and efficient resource allocation. The development of the hardware was completed in two stages: the IMU and multiplexer.

The MTDS IMU is a thumb-sized device containing Atmel AVR RISC 8-bit microcontrollers, InvenSense IMU3000 gyroscopes and Freescale MMA8451Q accelerometers. This research work is focused on understanding whether consumer-level electronics are sufficiently accurate to record human movements.

The MTDS multiplexer is a central node for acquiring motion data from the aforementioned IMUs. The IMUs are daisy chained to the multiplexer using a harness. As a whole, the system is wireless and connects to a computer over Bluetooth. While the multiplexer is able to interconnect up to twenty homogenous inertial measurement units, its testing was performed using the upper body configuration whereby IMUs are place on the arms, forearms, hands and torso.

1.4.3 Sensing Through Mobile Computing Technologies

As previously published in Pascu et al. [15] [16] and [19], this work's third contribution to knowledge is centred on investigating the relevance of mobile computing technologies, namely smartphones, to the fields of inertial motion capture and BSNs. This innovative concept demonstrates how inertial motion capture can be achieved using smartphone sensors and how BSN communications can be achieved using web technologies and the Internet. The smartphone is the most ubiquitous wearable computing technology. Most people have smartphones and, like IMUs, most smartphones enclose a gyroscope, an accelerometer and a magnetometer. This thesis discusses the design and development of a novel mobile application presenting a unique approach for constructing smartphone-driven BSNs that telecommunicate, in an omnidirectional manner, through web services. The BSN is designed to be used in small experiments and has many other application areas such as: medical science, activity tracking, emergency responses, road and traffic condition monitoring. To summarise, each network node uploads data, synchronizes data and distributes the result throughout the network. The mobile application extracts and processes motion data produced by the smartphone's sensors and communicates it across the network. The development of the mobile application is the result of porting and heavily modifying the software environment, previously introduced in section 1.4.1, to the Android platform.

The mobile application also implements a BSN remote control mechanism, which allows several smartphones to be controlled from one interface using event triggers [19]. An event trigger allows one smartphone's interface to remote control the actions of several other smartphones. Because this functionality is achieved using a web server and web services, an online control panel is also developed to control smartphones from a web interface. The concept of event triggers unifies several body-worn smartphones to emulate the functionality of a basic animation suit.

Synchronization refers to the process of gathering and merging multiple sets of motion data. The proposed solution is focused on using a centralised timeserver that updates each smartphone's clock. Clock readings can be used as reference points to identify precisely when each smartphone begins recording. Synchronization is important in creating an online smartphone-driven BSN that can merge the motion produced by separate devices.

1.4.4 Motion Cloud: A Repository and Gateway for Inertial Motion Data

This work's fourth contribution to knowledge is the Motion Cloud (Pascu et al. [19]), a solution that investigates the relevance of web technologies to inertial motion capture frameworks. Whether it's a single sensor or a network of sensors, the proposed architecture is highly extendable can be integrated with a wide spectrum of motion capture devices: animation suits, smartphones, inertial measurement units, pedometers, etc. To demonstrate its versatility, this thesis discusses several application areas where the Motion Cloud could be used as a library for motion data, a prototyping environment and data gateway for online BSNs and a database for storing activity data. The development of the Motion Cloud solution consists of three components: repository, gateway and web portal.

The repository is a large online database designed to store inertial data produced by software environments, mobile applications and drivers. Unlike existing solutions that allow users to upload motion capture files to online libraries, the Motion Cloud repository deciphers the data and constructs bespoke data models for each BSN in the form of object hierarchies.

While existing BSNs utilize multiplexers or software drivers to communicate data between nodes, the Motion Cloud gateway demonstrate the novel concept of constructing BSNs through web technologies. This concept is possible because of recent advances in telecommunication technologies (i.e. 3G, 4G) that facilitate data transfers between devices such as smartphones or IMUs. The gateway consists of versatile web services for uploading, downloading and streaming motion capture data. The summation of those services is an application-programming interface (API) designed to form a bridge between inertial sensing hardware and the Motion Cloud repository. The API can be used by downstream applications to access the repository's data.

The Motion Cloud can be interfaced with through a web portal. The web portal is an interface layer for accessing, modifying, visualising and exporting motion data stored by the repository. The web portal provides a quick overview of the data produced by a BSN as required for experimental research and other application areas.

1.5 Thesis Structure

This chapter introduces motion capture as a significant research topic and discusses its current problem areas. This work's contributions to knowledge are summarised and presented in response to the problem statement. The remaining portion of this chapter introduces the chapters to follow along with previous work that was influential to the research presented in this thesis.

Chapter 2 investigates fundamental motion capture concepts and application areas by generating a survey of pertinent literature. It emphasises significant problem areas in the field of motion capture and guides the research work presented throughout this thesis. A critical evaluation of the field is conducted to identify on-going research projects and the principal industry contributors. The chapter continues to discuss the individual stages of motion processing, from sensing the articulated movement of humans using inertial measurement units to reconstructing kinematic motion in virtual space.

Chapter 3 present the design and development of the software environment with emphasis on several core components: kinematics model and viewer, animation model and viewer, system engine and viewer, 3D rendering, DDK, LACOMA and BVHE. The chapter finishes by introducing the LACOMA, a computationally inexpensive solution for creating a better estimation of the kinematic model's anchor point (during gait) through weight distribution calculations.

Chapter 4 demonstrates how IMUs, which are commercially available, can be integrated with the framework. The chapter continues to present the design and implementation of MTDS while using existing technologies to form a baseline. The chapter discusses the conceptualisation, design, hardware development, firmware development, driver development and sensor integration. The chapter finishes with a comparative evaluation of the hardware.

Chapter 5 explores the relevance of mobile computing technologies to the field of inertial motion capture. The chapter covers the design and development of a mobile application for interconnecting smartphone devices to a server in order to establish omnidirectional BSNs. Smartphones are shown to be reliable test beds and prototyping environments for constructing BSNs aimed at experimental research and other application areas.

Chapter 6 introduces the Motion Cloud by discussing the design and implementation of the repository, gateway and web portal. The repository is presented as an extendable database model for storing and organising inertial data. The gateway is presented as an API for accessing the repository from: software environments, drivers, mobile applications, etc. The web portal is discussed in terms of interface design and overall usability.

Chapter 7 concludes this research by summarising the major issues raised throughout this thesis and the positive impacts of the proposed framework. Future projects, areas for future and development and extensions are also discussed.

1.6 Related Work

This section provides a chronological list of related work that the author has contributed to. The following four projects have provided both motivation and context for the developments presented in this thesis. Additionally, these projects were influential in forming a first-hand understanding of what problems affect the fields of inertial motion capture and BSNs.

1.6.1 eMove

eMove is a Technology Strategy Board (TSB) [20] funded research collaboration between the University of Sussex Centre for Computer Graphics and Animazoo [21], a motion capture hardware manufacturer and software developer. The focus of the project was to develop an upper body exoskeleton suit using one IMU, six potentiometers and two hand controllers featuring buttons, triggers and analogue sticks. The eMove suit [22] is designed for real-time digital puppetry [23] [24], interactive video games, theme parks, arcades, animation, etc.

The software suite developed for the eMove suit can be divided into three categories. The first category is concerned with drivers for accessing and streaming data directly from the hardware. The second category covers software applications for recording upper body motion for the purpose of animation. The third category presents software development kits for game engines such as Unreal Development Kit, Unity 3D and

Panda 3D. As a result, users have a strong foundation for utilising and further developing eMove suits.

eMove's ultimate goal is to deliver motion capture technologies to the masses. The problems faced with making exoskeleton hardware affordable, user-friendly and sufficiently robust to withstand everyday use are also applicable to the fields of inertial motion capture BSNs.

1.6.2 Motion in Place Platform

Motion in Place Platform (MIPP) [25] [26] [27] is an Arts and Humanities Research Council (AHRC) [28] funded research project with the aim of studying human-environment interactions in cultural heritage contexts. It brings together a cross-disciplinary group of researchers to create an understanding of how new technologies are beneficial in understanding the relationship between humans and their surroundings.

The project was centred on the findings of the Reading archaeologists [29] who uncovered the layout of Iron Age and early Roman buildings at the Silchester Insula XI. The author of this thesis used illustrated drawings and archaeological interpretations of those buildings to reconstruct the location in 3D. Motion capture data was recorded by equipping dancers, actors and archaeologists with inertial hardware to re-enact the daily activities of the historical inhabitants. They performed everyday tasks such as sweeping, cooking or getting water from a well. The data was recorded using Animazoo IGS motion capture suits and a laptop.

This research uncovered a list of problems concerned with using motion capture technologies outside the comfort of a recording studio. For example, using ultrasonic equipment outdoors for dead reckoning proved highly problematic due to windy weather conditions. The conclusions drawn from utilising motion capture technologies in the context of outdoor experimental archaeology indicate a need for less complicated, more robust, more portable and less encumbering BSNs.

1.6.3 Motion Capture in Forensic Psychology

This University of Sussex Centre for Computer Graphics undertook a forensic psychology experiment [30] [31] [32] together with an interdisciplinary team from University of Portsmouth Department of Forensic Psychology. The experiment's goal was to investigate the activity and behaviour of individuals burgling an urban household by studying human-environment interactions through motion capture technologies. The author of this thesis developed a realistic reconstruction of a household using a large set of reference pictures and measurements. Forensic psychologists used the reconstruction to create a virtual simulation of the motion capture data, which was subsequently used to analyse physical behaviour.

The experiment took two days to complete. On the first day, six university students were asked to burgle the household. On the second day, six previously convicted house burglars were asked to repeat the experiment in the same manner. Both groups were asked to navigate the environment and touch items they wish to steal. Their behaviour was recorded using an Animazoo IGS motion capture suit and a head-mounted camera.

The burglary experiment uncovered a list of technology limitations concerned with the overall usability, wireless connectivity and robustness of BSNs. The first problem occurred when test subjects were navigating the environment at a rapid pace and accidentally damaged sensors, causing the BSN to stall. The act of resetting the hardware in the midst of the experiment compromised the motion performers' psychological immersion. The second problem was concerned with the BSN requiring a constant wireless connection to a computer. On several occasions, the BSN disconnected due to distance and environment obstructions. The third problem was caused by the cumbersome nature of inertial systems as test subjects couldn't climb or enter the household through window openings without damaging the hardware.

1.6.4 Digital Hub

The Digital Hub [33] is a TSB [20] funded partnership between American Express and the University of Sussex with the aim of stimulating economic and business growth in the UK. The project is focused on developing technologies for micropayments, virtual currencies, rewards and loyalty schemes. The Digital Hub is exploiting mobile computing, social media, Internet of Things [34] and near field communication technologies to influence how people interact with virtual economies.

One particular project, entitled Fit2Gether, is relevant to the developments presented in this thesis. Fit2Gether is a cross-platform framework consisting of a mobile application and website for amalgamating activity data from users in the work environment. The framework's goal is to stimulate healthier lifestyles by rewarding users for physical activity. Data is captured using pedometers, thumb-sized devices enclosing accelerometers that produce rough estimations of steps taken, calories burnt and distance travelled. Rewards are used to stimulate users into changing their lifestyle from sedentary to active.

Pedometers are very similar to IMUs as they integrate both inertial sensors and microcontrollers in small packages. Modern pedometers are designed to upload data to a centralised repository using a set of web services. Data gets converted into activity readings while taking into account the user's weight, height, age, gender, etc. As demonstrated by this thesis, new-generation smartphones are abled to measure those properties using the in-built sensors. Fit2Gether's ability to store activity is similar to the Motion Cloud, which is a larger and more generalised solution aimed at amalgamating and organising inertial motion capture data in online repositories.

CHAPTER TWO

2 Motion Capture

2.1 Introduction

Motion capture is the notion of extracting data from motion sensors to reconstruct the movement of an animated being in virtual space. This chapter explores fundamental motion capture concepts and applications areas by generating a literature survey. To summarise, this chapter asks five basic questions: What is motion capture? What is motion capture used for? What technologies are available? How do those technologies work? How do users operate those technologies? By asking and subsequently answering these questions, this chapter reveals problem areas limiting the field of inertial motion capture.

This chapter's first aim is to classify motion capture as a new technology that is important to many applications areas, thus justifying it as the main research topic for this thesis. Novel applications areas for motion capture are discovered through an overview of on-going research. This includes: animation for game development and filmmaking, real-time motion capture and digital puppetry, biomechanics gait analysis, sport science, medical science, robotics and ergonomics.

The second aim of this chapter is to provide an overview of the main types of motion capture mediums (see Section 2.4) to identify strengths and weaknesses and to focus this research on a specific technology. This overview is supported by an evaluation of the motion capture industry contributors that identifies what systems are available commercially and what application areas they are used for (see Section 2.3). Based on this study, inertial sensing is chosen as the primary focus of research. The chapter continues to discuss the fundamental problems limiting inertial sensing and what attempts have been made towards solving those problems.

Aside from hardware, software applications play an important role in the usability of inertial motion capture systems. This chapter continues to evaluate what software applications exist, how they support motion capture technologies and how motion data is stored and transferred between users. The goal is to identify any standardisation-related problems affecting the industry and the field of research.

2.2 What is Motion Capture?

Motion capture is a general term that defines the process of constructing a virtual representation of motion using data obtained from real-life movement. As discussed in “Human Motion: Understanding, Modelling, Capture and Animation” [35], human motion is the primary target for motion capture due to its complexity and organic characteristics.

> “Motion capture involves measuring an object's position and orientation in physical space, then recording that information in a computer-usable form. Objects of interest include human and non-human bodies, facial expressions, camera or light positions, and other elements in a scene.” [36]

In the context of inertial motion capture, the act of recording real-life movement is a multifaceted sequence of procedures: data gathering, sensor fusion, pre-processing, post-processing, kinematic deployment, etc. One of the main goals of researching or developing motion capture systems is accuracy. In broad terms, accuracy is a measure of the difference between the originating real-life movement and the resulting virtual reconstruction.

While inertial motion capture suits are relatively new, the underlining principles are not. “The Mocap Book: A Practical Guide to the Art of Motion Capture” [37] covers the history of motion capture dating back to the 19th century. In recent years, inertial motion capture has become achievable and affordable because of technological advances focused on the miniaturisation of micro-electro-mechanical sensors (MEMS) and the availability of superior computational resources in small devices. Modern sensors are very robust and ubiquitous in devices such as smartphones, pedometers and animation suits.

In the context of this thesis, motion capture was chosen because it proves challenging and there are many problems still to solve. The act of recording and reconstructing motion happens very quickly. In less than a tenth of a second multiple sensors are interrogated to acquire motion data, motion data is processed and packaged, packages are sent to the computer through communication protocols, dead reckoning algorithms are applied and the result is visualised using a kinematic skeleton and a 3D engine.

2.3 Application Areas

Before discussing the principles behind software and hardware technologies, it is appropriate to understand what motion capture is used for. Motion capture has numerous application areas affecting our everyday lives [38]. Broadly, those application areas can be allocated to one of two categories: animation for digital entertainment and biomechanics. Animation is a result-driven process where motion is recorded to be modified at a later stage for filmmaking and game development purposes. For example, the animation of a computer-generated sci-fi robot presenting unrealistic proportions can be based on the motion recording of a human. The field of biomechanics is centred on the accuracy of the result in order to develop a better understanding of the original motion. For example, gait analysis requires data to measure the properties of human locomotion. In this context, how impressive the result looks is irrelevant. The following sections highlight the importance of motion capture by looking at the principal application areas, which can also be referred to as movement science, found throughout literature: animation, real-time motion capture and digital puppetry, medical science, gait analysis, sport science, robotics and ergonomics.

2.3.1 Animation

Animation is the dominant application area for motion capture through the affluent filmmaking and videogame industries. With advances in computer-generated graphics, more and more attention is given to the process of replicating motion in a virtual environment. Traditionally, the realism of animation was achieved through key frame animation and artistic talent but with an increasing demand for animations, manual replication of motion is too time-consuming and expensive.

Performance animation is principally found in the film industry where actors undertake the role of virtual characters to entertain an audience. “Understanding Motion Capture for Computer Animation and Video Games” [39] introduces the concept of performance animation whereby motion capture systems are used to capture the movement of motion performers (real-life professional actors).

In the past decade, the video game industry has surpassed the film industry in terms of budgets allowing for the development of more costly motion capture systems. In the context of video game animation, synthesis-by-example [40] is a term that refers to approaches for assembling pre-recorded sequences of animation. For example, a controlled game character might walk and then fight, requiring the two animation sequences to be blended seamlessly. Another application area for motion capture is facial animations. The most advanced facial animation solution is Rockstar's MotionScan [41] [42] technology that uses optical motion capture to generate a sequence of 3D face meshes. Those meshes are subsequently applied to character models sequentially to simulate organic-appearing facial expressions. This approach is a novel substitute to traditional kinematic motion reconstruction.

2.3.2 Real-Time Motion Capture and Digital Puppetry

In motion capture, animation data is recorded independently of its target use. After motion is recorded, post-processing methods prepare the data to be mapped to a digital avatar. Real-time motion capture is characterised by the motion performance and its virtual representation happening simultaneously.

Digital puppetry is a concept that refers to the motion performer as a digital puppeteer and the target avatar as a digital puppet. Digital puppeteers are real actors that perform to entertain an audience. To enhance the experience, lip-syncing technologies [23] may be used to animate the avatar's lips to match those of the performer. An example of digital puppetry is AnimaLive [43], a tool for creating virtual performances.

Real-time motion capture may also be used as a medium for interacting with virtual environments in the context of video games. As experienced first-hand throughout the eMove and Motion in Place Platform (MIPP) projects, motion capture systems are too expensive and fragile (due to the complicated nature of the hardware consisting of wearable cables and sensors) to be used on a daily basis. Real-time motion capture aimed at entertainment applications [24] is a novel concept that involves programming a motion capture system to feed real-time data into a game engine. Systems like Microsoft Kinect [44] have shown that motion capture systems can be used as an input device, through skeletal mapping or gesture recognition techniques [45] [46] [47].

2.3.3 Medical Science

Medical science is currently an underdeveloped yet important application area for motion capture. Recent literature suggests that recording human movements will create a better understanding of human anatomy, which benefits medicine and healthcare. This application area is possible as a result of advances in motion capture technologies allowing for more practical and affordable systems.

> “Motion capture systems have not been widely used in Parkinson’s disease research due to their high cost and lack of portability. However, recent advancements in portability and affordability have made various clinical applications possible.” [48]

Specific human movements can be examined to diagnose health-related problems and measure treatment responses. In a comparison between motion capture and traditional methods for medical examination, motion capture will produce a larger set of data consisting of 3D reconstructions and temporal information [49]. Therefore, physical examinations involving motion capture technologies provide medical professionals with valuable information that was previously unattainable. Inefficiency is caused by the complicated nature of using a motion capture system along with limitations in terms of hardware robustness. Inertial motion capture suits are likely to break while optical systems require a long setup procedure and a bespoke recording studio.

While optical systems are leading the field of medical motion capture, inertial technologies are becoming viable alternatives due to cost and versatility. For example, [50] presents the design and application of an inertial system to replace traditional rehabilitation in homes. The body-worn system measures the quality of the patient’s comportments and gives quantifiable scores that can be interpreted by a therapist.

Motion capture has potential in measuring musculoskeletal dysfunctions such as idiopathic scoliosis [51] and Parkinson’s disease [48]. In this thesis, Parkinson’s disease is considered as an application area for further research and development as published in Pascu et al. [15] [16].

2.3.4 Gait Analysis

Gait is a term describing the particularities of locomotion and is usually applied to human movement. Gait analysis concerns the activity of muscles and the symmetry of walking. It is applied to treat individuals with medical conditions that influence their ability to walk or their body's balance. In sports science, gait analysis helps athletes run more efficiently through better sports equipment (see Section 2.3.5) while preventing potential injuries.

To put into context motion capture's role in gait analysis we will consider three examples, as found in literature, in which motion data is used to analyse locomotion. In the first study [52] [53], an accelerometer-based system is used calculate acceleration patterns of the pelvis and head. From those patterns it was possible to determine whether a test subject is young and vigorous or old and frail, thus presenting the risk of falling and obtaining injury. In the second study [54], an optical motion capture system is used to determine whether a child's walking pattern differs from an adult's when overcoming an obstacle. Each test subject wears fourteen infrared markers as their motion is recorded. In the third study [55], the same optical system is used to determine the symmetry of walking in able-bodied elderly people to conclude asymmetries in the lower limbs over multiple gait cycles.

Motion capture technologies are inherently suitable for gait analysis as both fields focus on measuring and understanding motion. This application area is predominantly covered by optical systems [56] that are able to compute dead reckoning accurately. Dead reckoning is the process of measuring the combined distance travelled after a series of steps depending on stride length and foot placement estimation. Cloete and Scheffer [57] discuss the problems faced, in terms of experiment repeatability, of using inertial motion capture systems in the field of gait analysis. Clinical diagnoses require accurate portable motion capture systems that may be used by physicians. In this context, motion capture presents four problems:

1. Body-worn motion capture systems are cumbersome in nature and may influence the walking patterns of test subjects.
2. For gait analysis to be used in diagnosing, physicians require robust user-friendly systems.

3. Body joints are complex in nature and cannot be replicated accurately with traditional kinematic models.

4. It is difficult for systems, particularly inertial suits, to measure dead reckoning accurately for extended periods of time.

Current research projects aim to establish inertial systems and other motion capture technologies as powerful tools for gait analysis. An example research project is Outwalk [58] [59], a protocol designed to advance the kinematics of inertial and magnetic systems to measure thorax, pelvis and lower limb movements more accurately. Such research projects motivated the development of the lowest-anchor centre of mass algorithm (LACOMA) solution (Pascu et al. [18]) proposed by this thesis in Chapter 3.

2.3.5 Sport Science

Sports science is a developing application area for motion capture that covers the study of human motion during intense physical activity. Let us consider three usability scenarios that place motion capture in the context of sport science. First, motion capture is a medium that can measure and perfect an athlete's comportment during sport routines. For example, detecting and correcting the locomotive asymmetry of a runner may improve energy conservation allowing for a better sports performance. Second, motion capture can be used to aid the development of more comfortable sporting goods (e.g. running shoes, skis, snowboards, etc.), which reduce the risk of injuries. Third, motion capture can measure what forces athletes are subject to and create a better understanding of human behaviour under intense stress conditions.

Optical systems are firmly established in the field of biomechanics but due to the outdoors nature of sport, alternative motion capture solutions are being considered. What motion capture mediums are best suited to measure sport-related motion? Let us consider four studies focused on answering that question. The first two studies [60] [61] recorded the biomechanical data of freestyle snowboarding with the aim of reducing ankle joint injuries through development of better snowboarding equipment. The next two studies [62] [63] compare the accuracy of a full body inertial motion capture system with an optical video-based system in analysing the biomechanics of alpine skiing. These studies demonstrated that inertial systems are sufficiently accurate for this task.

2.3.6 Robotics

Motion capture has many robotics-related application areas [64] [65]. The design and engineering of autonomous machines is a process that benefits from the principles of natural motion (or biomechanics) on multiple levels. Motion capture can be juxtaposed with a branch of robotics that deals with artificial recreation and simulation of human motion. The robotic imitation of human movement [66] [67] [68] [69] can be achieved through motion in two ways. Firstly, limb movements could be mapped to robotic arms to effectively turn the robot into a puppet. Secondly, intelligent robots could use motion capture data samples to understand and perceptively replicate human behaviour.

Advances in modelling and calibration of inertial sensors and optical cameras can be beneficial in developing robots that interact with environments more intelligently. Optical motion capture algorithms can help robots understand their surroundings and perceive moving objects. Inertial sensors can be used to balance a robot's weight and improve locomotion abilities.

2.3.7 Ergonomics

Ergonomics is an application area that focuses on improving the interactions between humans and devices or environments. For example, the industry uses motion capture systems to prototype vehicle and aircraft interiors. Traditionally, ergonomic studies were completed through qualitative video observation, but motion capture technologies are becoming a viable ergonomics analysis alternative.

The study of ergonomics is also aimed at improving the design of body-worn equipment during movement to prevent muscle injuries and maximize comfort. For example, Optotrak systems have been used for ergonomic applications to assess personal carriage systems to improve backpack designs [70].

2.4 Motion Capture Mediums

There are different mediums through which motion can be recorded that vary in performance and cost. This section discusses the underlying principles of several motion capture mediums and looks at the main motion capture system manufacturers. An evaluation is made to summarise the strengths and weaknesses of each medium. Additionally, each medium is compared against the application areas previously identified to determine its versatility. The mediums discussed are: optical systems, inertial systems, exoskeleton mechanical systems and hybrid systems.

2.4.1 Optical Systems

A high-end optical motion capture system will involve a constellation of video cameras positioned around the target. A low-end system may use as few as three cameras positioned in a portable enclosure. The principle behind all optical systems remains the same: data is gathered in video file formats, computer vision filters are applied to the video stream and optical triangulation algorithms are used to deduce the position of the points of interest.

Optical motion capture may use reflective markers, pulsing light-emitting diodes (LED) markers or no markers to track points of interest in space.

- The reflective marker solution involves placing highly reflective indicators on the motion performer’s body. During the video recording, cameras shine light onto the markers to make them more visible.
- The pulsing LED solution uses infrared cameras to detect heat signatures.
- The marker-less solution estimates a person’s posture through computer vision algorithms. Advances in marker-less optical motion capture algorithms allow for the tracking of multiple interacting characters simultaneously [71].

Optical systems are highly accurate, noise-free and abled to compute dead reckoning. High-end optical systems are abled to track motion at very high frame rates, which is difficult with any other type of motion capture (especially inertial sensors which generally operate at frame rates ranging between 60Hz and 100Hz). Due to occlusion and light interference, optical motion capture requires a dedicated recording environment.

Vicon

Vicon Motion Systems and Peak Performance Technologies Inc. are two companies united under the Vicon [72] brand. This consortium produced a highly successful motion capture hardware and software company concentrating on highly accurate marker-based optical tracking systems. The Standard [73] is an annual magazine presenting Vicon's current projects and undertakings. A particular issue entitled "A Hopping Success for Outdoor Motion Capture" [74], explores the concept of outdoor motion capture, thus removing the need for bespoke recording environments. This concept could significantly increase the application areas for optical tracking systems. By combining highly accurate video cameras, like the Bonita shown in Figure 2-1, with motion analysis software, Vicon systems have a wide range of application areas.

Figure 2-1: Vicon Bonita cameras. **Source: [75]**

Optotrak

Optotrak Certus [76], shown in Figure 2-2, is an optical motion tracking system developed by Northern Digital Inc. [77]. The company specializes in designing and manufacturing optical motion capture systems for research application areas. The Certus combines a high definition position sensor and digital photogrammetry to compute motion data in real-time. Its three optical sensors are used for optical triangulation to determine the precise world-space positional coordinates of markers. Its cameras are able to detect the position and orientation of multiple moving objects in large open spaces by analysing data points at high frequencies. The accuracy of the Optotrak Certus made it useful to many research projects such as:

- *Medical Science:* studies focused on posture, balance and coordination include patient stabilization with trans-femoral amputation [78], idiopathic scoliosis [51], Parkinson's disease [79], etc.
- *Robotics*: comparison between human and machine operated industrial robots [80].
- *Ergonomics*: analysis of products (e.g. backpack designs [70]), systems, environments, tools, etc. [81].
- *Gait Analysis*: gait analysis of unimpaired elderly people [82] and obstacle avoidance in cluttered environments [54] [83].

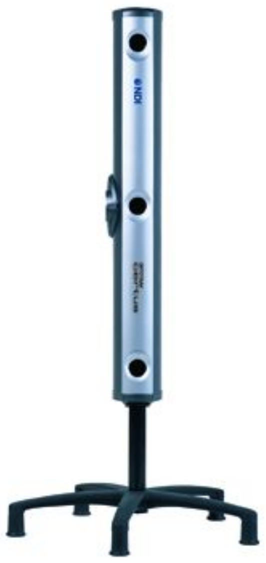


Figure 2-2: Optotrak Certus. **Source: [84]**

Microsoft Kinect

As shown in Figure 2-3, Kinect [44] is a consumer-level optical motion capture system developed by Microsoft for the Xbox 360 games console. Kinect's primary purpose is to allow users to interact with video games through gestures and spoken commands, thus removing the need for a game controller and creating a more interactive user experience. As an optical motion capture system, Kinect is a simple design using a video camera and a depth sensor. The video camera's primary purpose is to detect colour information for detailed gesture recognition. The depth sensor consists of a monochrome sensor and an infrared camera that work in tandem to detect the posture of the motion performer. Kinect's hardware is accompanied by image-processing software [85] [86] that estimates the world-space positions and orientations of body parts. Kinect has been used in many research projects because of its low-cost, software development kit (SDK) [87] and Robotics Developer Studio [88]. The popularity and low cost of the Kinect led to its use in many research projects.

- *Digital Puppetry:* controlling virtual characters in video games.
- *Medical Science:* the gamification of patient rehabilitation [89], the training of nurses in performing patient transfers [90], studies of body joint movements [91], systems for training of factory workers to lift safely [92], etc.
- *Robotics:* recreating human motion through electro-mechanical skeletons [93], and robots [94].
- Gait Analysis: capturing and evaluating of ambulatory behaviours [95] [96] [97] for experimental research.

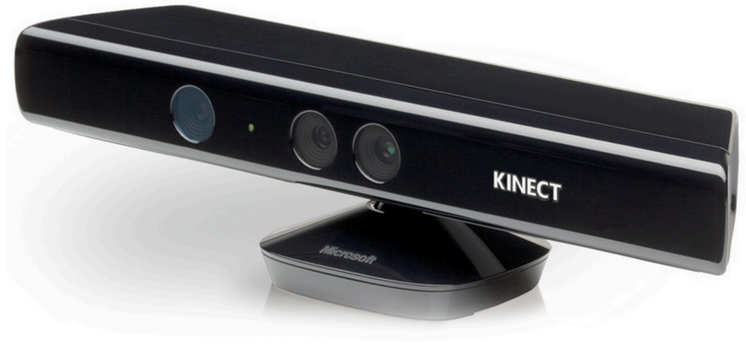


Figure 2-3: Microsoft Kinect. **Source: [44]**

2.4.2 Inertial Systems

Inertial motion capture uses sensor technology, motion processing algorithms and kinematic models to record movement. Inertial motion sensors such as gyroscopes, accelerometers and magnetometers are used to identify three-dimensional world rotations. A virtual representation of the performer is constructed by applying these rotations to a kinematic skeleton. For a system to identify the performer's full body movement, a sensor unit must be placed on each major body part: limbs, torso, neck and head. The inertial sensors may be worn using straps or sown into clothing depending on size. Most motion capture systems interconnect approximately twenty sensors through a central hub commonly referred to as a multiplexer. Inertial technologies benefit from a wider range of application areas as they are portable and may be used outside the comfort of recording studios. Unlike optical systems, inertial motion capture suffers from sensor-related inaccuracies such as drift and noise.

XSens MVN

XSens [98] is a manufacturer and supplier of MEMS inertial sensor products and technologies. MVN [99] is a department of XSens that focuses on creating motion capture suits for animation and biomechanics. Their latest product is an inertial motion capture suit featuring seventeen in-house built MTx sensors strapped to clothing. The MVN suits, shown in Figure 2-4, have made their way into the filmmaking and video games industries. XSens products target several application areas:

- *Animation*: character tracking for blockbuster films and video games.
- *Medical Science:* analysing human movement in clinical trials [100] [101] assessing the frailty of elderly people [102], measuring trunk and gait parameter [103], etc.
- *Sports Science:* motion analysis of human behaviour during sport activities such as snowboarding [60] [61] and alpine skiing [62].
- *Gait Analysis:* capturing and evaluating ambulatory behaviours [104] [105] [106] for experimental research.

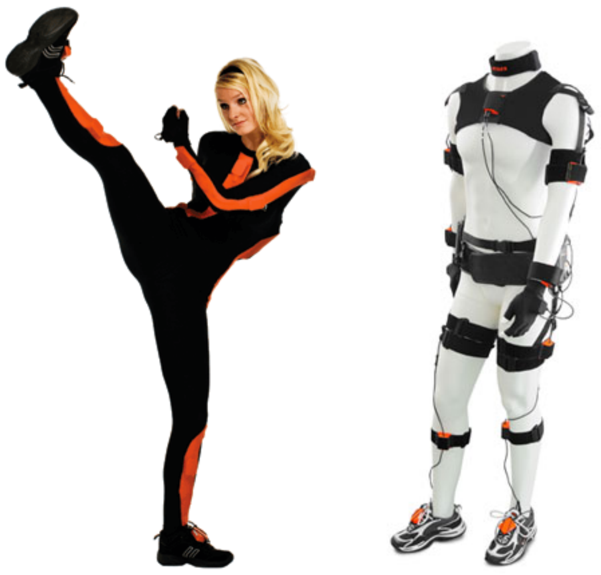


Figure 2-4: XSens MVN full-body suit. **Source: [99]**

Animazoo

Animazoo is an animation-driven motion capture software and hardware developer. It has a wide range of motion capture solutions that vary in price and performance. As shown in Figure 2-5, the most power Animazoo system is the IGS 180, an inertial motion capture suit interconnecting eighteen MEMS inertial measurement units to a wireless multiplexer. Application areas for Animazoo systems include:

- *Animation:* character motion tracking for films and video games.
- *Digital Puppetry:* animating in real-time virtual characters with the added option of lip synchronization through AnimaLive [43] software.
- *Ergonomics:* measuring the interaction of humans with vehicle interiors.

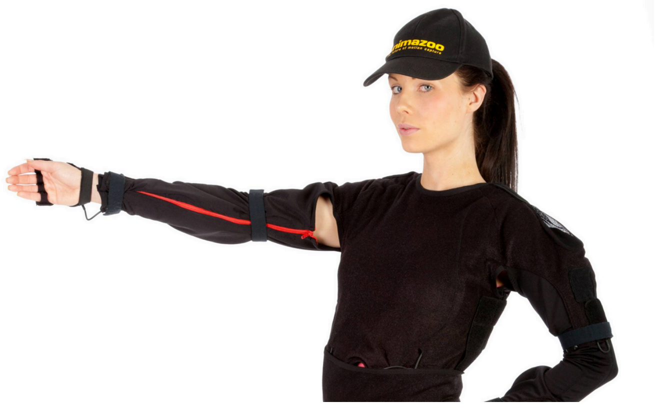


Figure 2-5: Animazoo IGS190 full-body suit. **Source: [107]**

2.4.3 Exoskeleton Mechanical Systems

Exoskeleton motion capture systems are body-worn prosthetic structures that can be worn on top of clothing. Typical hardware consists of extendable rods connected through potentiometers. The rods mimic the user's physical skeleton while the potentiometers record the rotation of joints. Potentiometer readings are converted into rotations and applied to kinematic models as skeletal motion. Animazoo holds patents for exoskeleton systems, making it the only vendor for these technologies.

eMove

The eMove project [22] originated from a Technology Strategy Board funded partnership between the University of Sussex and Animazoo. The project produced a prototype exoskeleton personal motion sensing system entitled eMove and now sold commercially as The Wing and Gypsy 7. The system was originally derived from the Animazoo Gypsy 5 prototype. As shown in Figure 2-6, full-body and upper body suits are designed to record torso and limb movements. Exoskeleton's biggest downfall is the fragility of the complicated hardware.

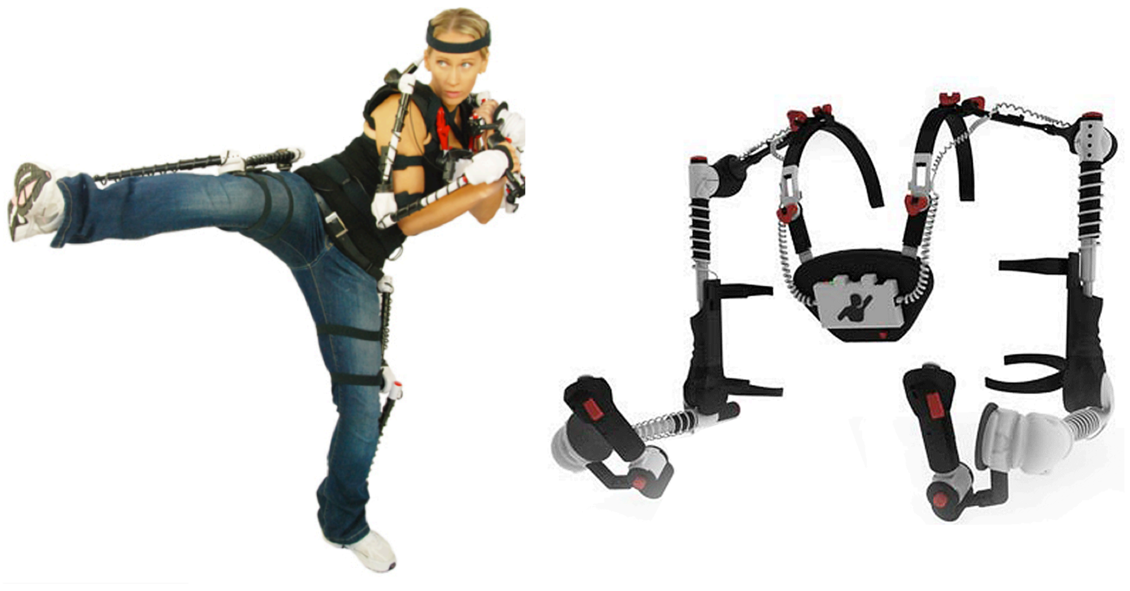


Figure 2-6: Early eMove prototypes: Full Body (left) and Upper Body (right). **Source: [108]**

2.4.4 Evaluation of Motion Capture Systems

Table 2-1 provides a brief summary of the advantages and disadvantages of motion capture mediums.

Table 2-1: Advantages and disadvantages of motion capture mediums.

Software	Advantages	Disadvantages
Optical	- accuracy - recording frequency - robust - inherently able to compute dead reckoning	- cost - require dedicated recording spaces - sensitive to lighting conditions - occlusion-related limitations - not usable outdoors - not usable in confined spaces - not portable
Inertial	- accuracy - cost - usable outdoors - usable in confined spaces - portable	- slower recording frequency (compared to optical) - fragile - dead reckoning - magnetometers are affected by magnetic interference
Exoskeleton	- cost - usable outdoors - portable	- accuracy - recording frequency - cumbersome - fragile - dead reckoning - not usable in confined spaces

Optical systems have many benefits in terms of accuracy, high recording frequency, durability and dead reckoning. At the same time, optical systems present many drawbacks in terms of cost and usability. While there are studies attempting to make optical technologies usable outdoors, optical sensors are likely to be confined to specific recording environments that present ideal optical lighting conditions. Optical systems have presented a large number of application areas in research-related fields. Exoskeleton systems currently play a very small role in the field of research. While cost is an advantage, accuracy, fragility and the cumbersome nature of the hardware remain major problems. The question then arises: why wear prosthetic hardware when inertial suits achieve the same task more accurately? While cost is a factor, it is worth mentioning that modern MEMS are decreasing in price while improving in performance.

Inertial systems provide a balance between benefits and drawbacks. Suits are highly portable and usable indoors, outdoors and in confined spaces. While accuracy can be problematic, modern MEMS sensors are sufficiently accurate for a wide range of application areas. Sensor-related problems such as drift and noise are being resolved through better hardware, firmware and software.

2.5 Inertial Measurement Units

The smallest and most important element of an inertial motion capture system is the inertial measurement unit (IMU). As the name would imply, IMUs are sensors designed to measure rotational or gravitational motion. IMUs generally contain a gyroscope, accelerometer and magnetometer.

An IMU is a device encapsulating one or several sensors along with a small processing unit. For example, InvenSense sells a nine-axis sensor entitled MPU9150 [109] containing a gyroscope, an accelerometer and a magnetometer. IMUs (particularly those designed to be used in research) will generally contain an additional communication mechanism (e.g. a Bluetooth emitter, USB connector, etc.).

Modern IMUs contain on-board processing units that run digital filters and sensor fusion algorithms to improve accuracy and provide developers with a usable output. Aside from motion capture, IMUs have been used to in aeroplanes, space rockets and military missiles for the past decades. More specifically, the technology found in IMUs today was derived from those application areas. With hardware miniaturization, IMUs have become ubiquitous in many devices such as smartphones or tablets.

But why do IMUs require specifically a gyroscope, accelerometer and magnetometer? The gyroscope cannot measure rotations accurately by itself due to drift and other inaccuracies because it does not have a reference point. A gyroscope produces positive rotations even when motionless. Accelerometers and magnetometers are used as reference points by the gyroscope. More specifically, the accelerometer measures the gravitational force (vertical axis) while the magnetometer measures geographical north (horizontal axis). The third axis can be computed from the two to give the gyroscope an orthogonal reference point. This process is referred to as sensor fusion and is further discussed in Section 2.5.3.

2.5.1 Gyroscope

The gyroscope is the core component of any inertial motion capture system. Gyroscopes are used to measure the orientation of the device on which they are mounted. Depending on whether the chip contains an inbuilt processor, orientation data can be expressed as angular speeds, rotations or even quaternions. There are two types of gyroscopes that are popular: micro-electro-mechanical sensors (MEMS) and piezoelectric. Consumer devices such as smartphones contain MEMS gyroscopes that are ubiquitous and very robust. These microchips are becoming increasingly more accurate while their cost is decreasing. Consumer-level MEMS chips are now sufficiently accurate for inertial motion capture applications (e.g. measuring the biomechanical data of the human body). "MEMS Vibratory Gyroscopes: Structural Approaches to Improve Robustness" [110] provides a good introduction to the history of gyroscopes and how they function. "Practical MEMS: Analysis and Design of Microsystems" [111] provides an overview of MEMS gyroscopes and other sensors by looking at examples of existing hardware architectures.

2.5.2 Sensor Attributes

The following list summarises the key attributes of an IMU that can be used to measure performance. There is no standardized way of expressing how good a motion sensor is as manufacturers advertise their sensor products in dissimilar ways.

- *Number of Sensing Axis:* Gyros can be single, double or tri-axial measuring angles in one, two or three axis. Fundamentally, tri-axial sensors use the components of three single-axis sensors oriented orthogonal to one another. Modern inertial motion capture generally uses tri-axial MEMS sensors.

- *Full Scale Range:* The range of a sensor, measured in degrees per second, specifies the maximum angular change that can be sensed in a set time interval.

- *Sensitivity:* Sensitivity, measured in least significant bit (LSB) per degree per second, specifies how small a movement can be for the sensor to detect. It is calculated by looking at how the LSB oscillates for a one-degree rotation in one second. For example, if a gyroscope rotates one degree in one second, the last bit of the angular reading that is modified is indicative of sensitivity.

- *Bandwidth:* Bandwidth, measured in hertz, is the maximum frequency of readings that can be made per second. A sensor is generally more accurate if it produces data more frequently.

- *Shock Toleranc*e: All sensors contain moving parts that may be subject to breaking. Consumer-level devices can be damaged if dropped and require replacement. Shock tolerance is a measure of durability. In military and aviation applications, sensory devices must function correctly while enduring large gravitational forces. With advances in MEMS technologies, most modern consumer-level digital sensors are robust and can withstand very large shocks.

- *Calibration:* Calibration is the process of compensating a sensor's output to reflect an accurate reading of the motion. For example, a gyroscope's output may detect 91 degrees of motion when rotated only 90 degrees. The 1 degree difference needs to be detected and accounted for at the firmware level. Even though sensors may require recalibration after extensive use or if damaged, calibration has become uncommon in modern consumer-level electronics as, components are less expensive to replace than to calibrate.

- *Drift*: Drift, also known as Angular Random Walk (ARW), is a sum of gyroscope inaccuracies over longer periods of time and is a measured in degrees per hour. When a gyroscope is held motionless, its rotation gradually changes without actual movement. For this reason, IMUs contain additional accelerometers and magnetometers to correct noise.

- *Noise:* Noise is characterised by random fluctuations within a sensor's data. Sensors produce minute highpoints and depressions that oscillate above and below the real values. Noise can be corrected through filters or damping algorithms. Without filters, accelerometers and magnetometers produce a large amount of noise.

- *Bias Error*: Similar to drifting, the bias error is the distortion of outputs over a period of time. The bias error is semi-predictable as the sensor is tested against different internal and external factors. For example, the sensor heating up to its functioning temperature upon start-up is a factor. These factors must be understood and accounted for by defining an offset value.

- *Size*: The size of the sensor is critical in practical applications. A smaller sensor is likely to consume less power and produce less heat than a larger sensor. While smaller sensors are generally more desirable, they are more difficult to solder onto circuit boards manually, which makes the prototyping of electronics difficult for the average user.
- *Power Consumption*: To maximize battery life, wireless devices require low-power sensors. For example, low-power accelerometers can be found in commercial pedometers.
- *Output Formats*: Because sensors contain inbuilt processing units, they can produce data various formats. For example, a gyroscope may outputs data as angular speeds, angular rotations or even quaternions.
- *Sensor Fusion and Filters*: Sensor fusion and filters are features that enhance performance. A small processing unit must be placed within the sensor to process sensor fusion algorithms and filters. Because of this, modern sensors are more accurate, reliable and noise-free.

2.5.3 Sensor Fusion

Performance in motion capture is defined by accuracy: a measure of how close the recorded motion reading is to the actual motion. Accuracy can be expressed in two ways: reproducibility and reliability. Reproducibility is a measure of how close a recorded motion is to its true value while reliability is a measure of consistency over lengthier periods of time. Considering that IMUs are required to read motion correctly and dependably for extended periods of time, sensor fusion becomes key in maximizing performance in new motion capture systems.

Sensor fusion is the process of combining multiple data sets to conclude a more accurate reading of motion. By fusing the data of an IMU containing three separate sensors, it is possible to conclude readings that are superior to those produced by each sensor individually. Generally, sensor fusion algorithms are mathematical solutions that focus on improving two key properties: signal noise and drift. Sensor fusion can be described as being either heterogeneous or homogenous.

In homogenous sensor fusion, an IMU will contain several identical sensors that are implemented to work in parallel to produce enhanced motion readings. For example Gyro-Free Inertial Navigation System (GFINS) [112] is a sensor unit that implements six accelerometers to work congruently. The result is the averaged output of all the devices. Another example is EcoIMU [113] an IMU that integrates two accelerometers in a similar fashion.

In heterogeneous sensor fusion, an IMU will contain different sensors where each sensor has a distinct purpose. It is a complementary solution whereby each sensor has a unique strength that can be used to reduce another sensor's limitation. As previously mentioned, a typical IMU will contain a gyroscope, an accelerometer and a magnetometer. The gyroscope's strength is its ability to sense noise-free motion at high frequencies. The accelerometer and magnetometer's strength is to produce drift-free data, which can be used as a reference. Consequently, the gyroscope's noise-free data can be compensated with the accelerometer and magnetometer data to reduce drift.

2.5.4 Kalman Filter

The most popular algorithm for sensor fusion is the Kalman filter, a mathematical solution for reducing random variations in motion. The Kalman filter gathers inaccurate data from multiple sensors and produces a reading that is closer to the real motion. "Introduction to Random Signals and Applied Kalman Filtering" [114] provides an insight into the random process theory and introduces the Kalman filter with a strong emphasis on its applications and implementation. To put the Kalman filter into the context of inertial motion capture, [115] provides an example of a design and implementation of a Kalman filter that uses quaternion mathematics.

2.6 Body Sensor Networks

Body sensor networks (BSN) are constellations of sensing devices that measure key properties of the human body [107] [116]. In the context of motion capture, BSNs use IMUs containing gyroscope, accelerometer and magnetometer sensors. “Body Sensor Networks” [117] provides an overview of sensor networks with emphasis on advances in wireless systems and discussions on application areas in the field of medical science.

The human body can be represented as a hierarchy of skeletal segments where each segment corresponds to a bone in the body. Depending on age, the human body has between 200 and 300 bones that fuse with age. In computer graphics, the skeleton can be represented, in a more simplistic manner, as a kinematic model. A typical BSN will use as much as 20 IMUs placed strategically on each major body part. For example, the Animazoo IGS 180 uses 18 IMUs placed on the limbs, hands feet, torso, neck and head.

Smaller IMUs can be sown into clothing while heavier IMUs may be worn using elastic straps. While moving, sensors will slide in relation to the skin and cause inaccuracies. At the same time, wearing IMUs that are strapped tightly will constrict movement. Weight is also an important factor as a heavy IMU will move more in relation to the body than light IMU. The problem area concerned with the physical act of wearing IMUs can be referred to as sensor distance noise [118] and represents a major source of inaccuracies for inertial motion capture systems.

Although inertial motion capture suits are a relatively new development, the fundamental concepts behind BSNs have been exploited in other application areas outside the context of human motion. For example, hospitals use sensor networks to monitor the physical properties of patients (e.g. pulse, heart rate, blood pressure, blood oxygenation, etc.). The act of measuring subtle movements can provide important information about a patient’s condition. Mercury [119] is an inertial BSN used to supervise patients with severe cases of epilepsy or Parkinson’s disease. This system consists of eight IMUs placed on the body whereby each sensor can be worn on the arm, forearm, calve or shin. The sensors are connected wirelessly to a laptop computer to avoid problems caused by cables interfering with the patient’s natural motor functions.

To further understand the concept of BSNs, the following sections discuss four key properties: networking, homogenous/heterogeneous BSNs, directionality, modularity and computational centricity. These concepts are further discussed in Chapter 3 and Chapter 4 by creating a dedicated motion capture software environment for extracting data from sensors.

2.6.1 Networking

Networking is concerned with gathering data from the IMUs and merging the result to produce a complete recording of the body. Networking involves establishing connections, interrogating and synchronizing the data produced by each device. There are two types of sensor networks: wired [107] and wireless BSNs (wireless BSNs are commonly referred to as wireless body area networks (WBAN) [120]). Wired BSN have fewer power usage constraints while WBANs are more practical. Data is sampled from each network node at specific time intervals. Once all the IMUs are interrogated, synchronization algorithms are applied to merge the data and produce a complete recording of the body's motion.

2.6.2 Heterogeneous and Homogenous BSNs

BSNs can be heterogeneous or homogenous depending on the variety of sensor used. Most inertial motion capture systems are homogenous and use an array of identical sensors. For example, the ETH Zurich Sensor Hardware system [118], which is based on the Smart-its platform [121], extracts data from 48 accelerometers that can be placed on the body using straps. Similarly, the Lancaster Multi-Accelerometer Platform [118] [122] embeds 30 accelerometers in a pair of trousers and a lab coat.

2.6.3 Directionality

Directionality represents the flow of data throughout BSNs and it can be directional or omnidirectional. Most inertial motion capture systems are directional whereby the communication with the nodes is the process of extracting data. Aside from basic commands, no motion data is sent to the IMUs. However, a sensor may benefit from knowing what its surrounding nodes are measuring. This concept is further evaluated in Chapter 5 by creating an omnidirectional smartphone-driven BSN. Figure 2-7 illustrates an omnidirectional upper body sensor network with 8 IMUs. The IMUs are daisy chained to a multiplexer as the data is gathered from each node.

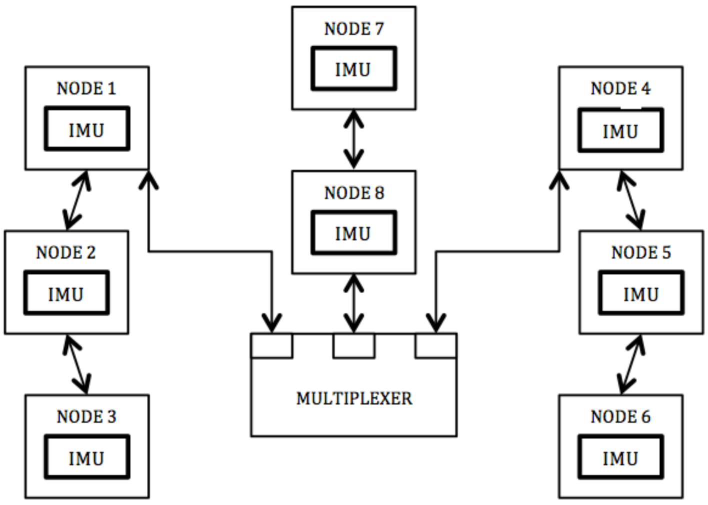


Figure 2-7: Omnidirectional body sensor network.

2.6.4 Computational Centricity

A BSN can be classified as being hardware or software-centric depending on whether motion processing is achieved at the hardware or software level. Typically, an IMU is software-centric if it outputs raw unfiltered data in the form of angular speeds. Software is then used to compute rotations, apply sensor fusion algorithms and filters. A hardware-centric IMU will contain in-built microcontrollers that are abled to achieve these computations within the hardware.

Depending on application areas, there are advantages and disadvantages to each approach. It is much more convenient for researchers willing to integrate sensors within their software to have hardware that can compute motion. Hardware that computes motion requires more powerful processing units, which in turn affect power consumption, size and cost.

Hardware-centricity is becoming a key requirement for new inertial motion capture systems with the aim of making hardware more autonomous. It is more efficient to distribute the workload between a suit's microcontrollers than to send it to a computer. This concept is further discussed in Chapter 5 through the design and implementation of an omnidirectional smartphone-driven BSN.

2.7 Kinematic Motion Reconstruction

This section introduces the theory behind rotational models, kinematic models, collision detection and dead reckoning. These stages of computation take place at the software level once all the data has been gathered from the BSN.

2.7.1 Rotational Models

There are multiple methods for computing and storing the rotations produced by IMUs. A rotational model describes a mathematical representation of rotations that, in the context of inertial motion capture, can be used to express a kinematic model. There are multiple rotation models: Euler, quaternion, rotation matrices, Rodrigues and invariant representations using rotation tensors. The two rotational models most often found in computer graphics are Euler rotations and quaternions. Inertial motion capture software applications use a combination of both models. These models, their mathematical operations and conversion equations are further discussed by [123].

The Euler model represents transformations as an ordered sequence of three rotations corresponding to the rotational yaw, pitch and roll. The Euler model is the only intuitive solution for representing rotations. For example, plotting Euler rotations on a graph provides an interpretable visualisation of the motion. However, this model becomes problematic because of gimbal lock. Representing a rotation within a three-dimensional gimbal can simulate this problem. Gimbal lock occurs when one degree of freedom is lost as two of the three discs become parallel, thus restricting the system to a two-dimensional configuration.

A simpler solution to the *gimbal lock* problem is quaternion algebra. A quaternion is a set of four values within four-dimensional (4D) vector space consisting of a real number and three imaginary values. Quaternions are generally used in computer graphics to rotate 3D objects in space and produce animations. Two quaternion rotations can be summed through quaternion multiplication, which is noncommutative. Quaternion multiplication has many benefits in terms of performance as it can be broken down into mathematical multiplications, additions and subtractions. Those simple operations are significantly faster to process by computers than trigonometry (which is required for the Euler model). The theory behind quaternions is further discussed in Appendix A.

2.7.2 Kinematic Models

Once all rotational data is gathered from an inertial motion capture system, a kinematic model is used to reconstruct a virtual representation of the performer. The kinematic model can be used to create 3D visualisations of the motion. As shown in Figure 2-8, kinematic models are skeletal hierarchies containing rotational and positional constraints.

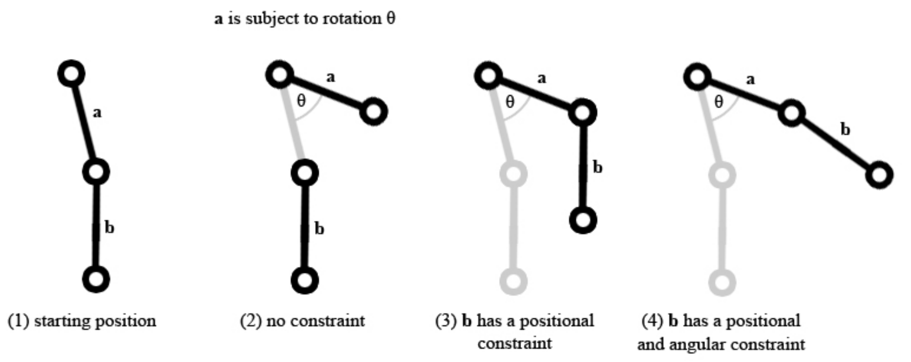

a is subject to rotation θ
a
b
θ
(1) starting position
(2) no constraint
(3) b has a positional constraint
(4) b has a positional and angular constraint

Figure 2-8: Position and rotation constraints in kinematic models.

Kinematic models are used to visualise the movement of a motion performer as an articulated skeleton. That skeleton can be assigned to animate a virtual character. Aside from visualisations, kinematic models are also important in the context of dead reckoning (see Section 2.7.4). Figure 2-9 illustrates a rendered kinematic model performing three gestures: walking, dragging feet and kneeling.

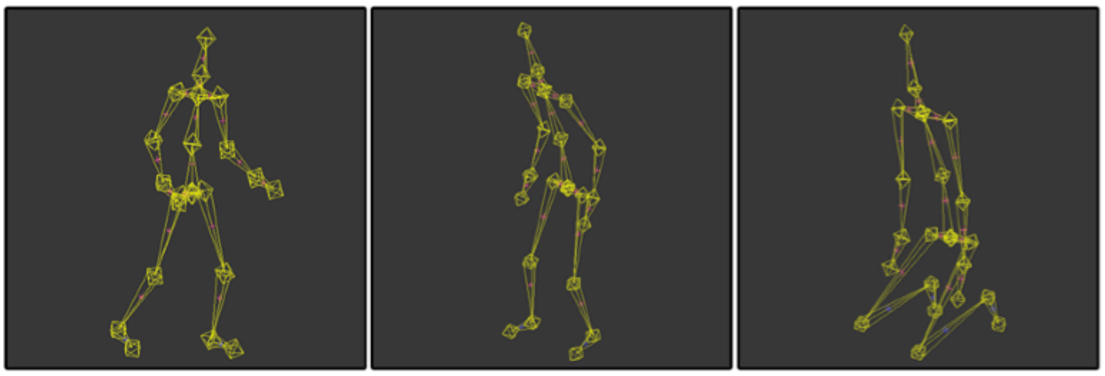


Figure 2-9: Kinematic models playing back motion. **Source: Pascu et al. [18]**

2.7.3 Collision Detection

Collision detection, sometimes referred to as collision avoidance, is the problem of computing the intersection between two or more objects. Collision detection is used in motion capture kinematics to compute an estimation of horizontal displacement. This process is also referred to as dead reckoning and is further explained in the next section. For inertial motion capture systems, collision detection is used to create a realistic simulation of foot placements.

In reality, the human foot endures complex musculoskeletal deformations. Considering that most motion capture suits only use one sensor per foot, those deformations are too intricate to record. Basic implementations of collision detection in video games and animation can increase the believability of the virtual character. When walking, collision detection can cause *foot-skating* whereby the foot moves when parallel to the ground. The concept of *foot-planting* [124], which describes feet stationary when balancing a character's weight, could solve this problem. When the foot is planted it experiences a floor constraint. When the foot is lifted, its position and rotation must be interpolated between its planted state and its unconstrained state.

Behavioural studies, particularly those concerned with posture, balance and coordination require elaborate kinematics where collision detection is used to simulate pivotal constraints between bones. Anatomy-accurate musculoskeletal systems implementing advanced collision detection methods [125] may prove beneficial to the field of biomechanics. A more accurate and realistic musculoskeletal system can be achieved through by creating a kinematic model that utilizes advanced joint interactions [126].

2.7.4 Dead Reckoning

Inertial motion capture systems contain no inherent sensors that measure horizontal displacement. As a result, walking algorithms must be applied to the kinematic model to compute an approximation of horizontal displacement. For example, taking a step forward will propel the kinematic model in the direction of travel. Also referred to as spatial positioning, dead reckoning is the process of computing horizontal displacement for longer periods of time. For example, taking a set of steps will translate the kinematic model in space by a certain amount. Dead reckoning algorithms can be used to calculate that amount during processing or post-processing stages of motion processing. Additionally, peripheral hardware can be used to create a more accurate estimation of horizontal displacement.

Dead reckoning data can be added to the recording manually using animation software. However, translating the kinematic model manually is a time-consuming task that will produce inaccuracies. Physics engines, such as those found in game engines, can be used to add physical properties to the kinematic model to synthesize horizontal displacement. However, physics engines are computationally expensive during real-time motion capture.

A popular approach for determining horizontal displacement is the lowest-point algorithm [127]. This computationally inexpensive algorithm produces a rough approximation of the distance travelled by selecting the lowest kinematic segment and applying planar collision detection.

Figure 2-9 illustrates the functionality of the lowest-point algorithm as a sequence of kinematic computations. The supporting limb of the kinematic model is often referred to as an anchor point because the entire skeleton moves in relation to that point due to positional and rotational constraints. With each step taken, the anchor swaps between feet. As the anchor moves in space, the skeleton is propelled forward.

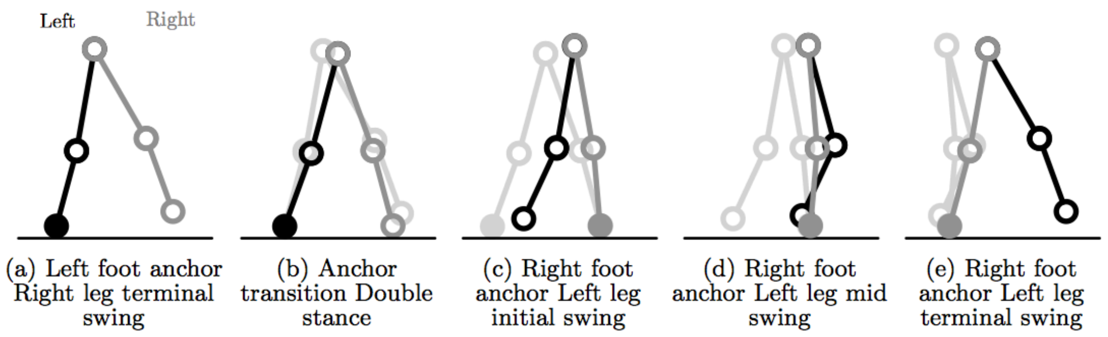

Left
Right
(a) Left foot anchor Right leg terminal swing
(b) Anchor transition Double stance
(c) Right foot anchor Left leg initial swing
(d) Right foot anchor Left leg mid swing
(e) Right foot anchor Left leg terminal swing

Figure 2-10: Lowest-point algorithm. **Source: [127]**

If the skeleton is calibrated accurately, this algorithm will produce reasonable horizontal displacement. However, the human body is much more complicated than a kinematic skeleton. Representing the human body as simple kinematic pivots will lead to inaccuracies.

The lowest-point algorithm only works when the test subject is taking clear steps where the detachment from the ground of each foot is detectable. More specifically, the algorithm is likely to miscalculate which kinematic joint is the correct anchor if the person is dragging their feet.

Dead reckoning may be achieved using peripheral sensors. The most common two types of peripheral sensors are ultrasonic sensors and video cameras. The act of integrating additional sensors, which are not motion sensors, in a BSN can be referred to as hybrid motion capture. While hybrid motion capture is highly accurate, there are many disadvantages, which limit the systems in terms of practicality and versatility. A BSN that uses peripheral sensors is no longer portable and requires a designated recording studio. In the case of ultrasonic sensors, outdoor recordings are problematic due to noise interference (e.g. from the wind).

2.8 Motion Capture File Formats

As previously mentioned, animation is the most thriving sector of the motion capture industry consisting predominantly of filmmakers and game developers. This prosperous industry drives motion capture technologies forward by establishing new recording studios. As the industry expands, more studios are focusing on creating new methodologies for storing, transferring and trading motion capture data. Those methodologies are not yet standardised causing cross-compatibility issues between systems and software environments. "Working With Motion Capture File Formats" [128], [129] and [130] provide overviews of what motion capture data is and what file formats are used to contain it.

> "…motion capture has a significant weakness due to the lack of an industry-wide standard for archiving and exchanging motion capture data. It is difficult for animators to reuse and exchange motion capture data with each other." [131]

Motion Capture Markup Language (MCML) [131] is a proposed framework that aims to solve the data compatibility problem by homogenizing numerous file formats. MCML is built using the eXtensible Markup Language (XML) and aims to simplify file format conversions. Additionally, this framework introduces the concept of a motion database to organize and manage numerous recordings. There is a general need for an industry-wide standard for storing motion capture data. This thesis proposes that web technologies could be used to removing the need for local file storage altogether by storing motion capture data in online databases.

All motion capture systems output similar data consisting of either angular rotations or positional coordinates. The following terms are frequently used when describing motion capture file formats:

- *Bones and Joints:* The building block of a 3D skeleton representing a segment corresponding to a limb section. For inertial systems, bones represent body-mounted sensors having an origin, a length and an orientation.
- *Skeleton and Skeletal Hierarchy:* The bone topology defining the shape or arrangement through which a motion-captured character is defined in virtual space. A topology will depend on what system is used and how many sensors are interrogated to gather data. A skeletal hierarchy relies on bone dependencies to interlink its constituent segments.
- *Motion Frame:* A snapshot recording of the performer's posture.
- *Channel:* Data structure storing the stream of angular readings for one sensor.

Most motion capture file formats store both angular readings and kinematic definitions in the same file. The three most popular file formats are: Biovision Hierarchy (BVH), Biovision (BVA), Acclaim Motion Capture (AMC) and Acclaim Skeleton Format (ASF). These formats are further discussed in the context of software compatibility. The biggest problem with these formats is that they contain no motion capture system specific data (e.g. information how the system was configured to produce the data). In essence, these formats represent different methods for expressing character motion with disregard for the recording technology.

2.8.1 BVH Format

BVH is an easy-to-parse ASCII format that stores motion capture angular readings and positional offsets. BVH appears to be the most popular motion capture file format in animation packages. For that reason, it is implemented in the software environment proposed by this thesis. Additionally, it is the only format that includes a skeletal hierarchy template and motion data within the same file. The syntax is divided between the file's header and body.

2.8.2 BVA Format

The BVA file format stores angular readings and positional offsets within segment definitions. Unlike BVH, BVA does not support hierarchy definitions making it impractical for certain applications. BVA can only be used in scenarios where the software environment has a preconfigured kinematic model. In an ideal scenario, the user should be able to decide which kinematic model to use and to specify what system is being used to record motion. As proposed by this thesis, these configuration attributes could be specified in a more comprehensive file format.

2.8.3 AMC/ASF Format

The AMC/ASF file format is widely supported by applications and considered to be the most comprehensive. It contains both skeleton hierarchy information and motion capture data within two file types: Acclaim Skeleton Format (ASF) and Acclaim Motion Capture (AMC). ASF stores the skeletal definition while AMC stores the motion capture data. In summary, Acclaim has developed a very complete motion capture file format that serves the task of storing both motion capture data and skeletal information very well. Although the Acclaim format is widely used, it is not the most concise or easy-to-parse solution of the three presented. Its functionality is very similar to that of a BVH file yet the syntax is more complicated.

2.9 Supporting Software

This section examines the compatibility between motion capture systems and software applications. Inertial motion capture systems are regarded as new technologies and therefore have little support from popular computer software applications. Because there are no bespoke software applications for extracting data from sensor devices, software support can only be analysed in terms of what motion capture file formats are supported by which software application. Because character animation is regarded (by the industry) as the main application areas for inertial motion capture, the main software applications that support the aforementioned file formats are tools for animation. This raises the question: can animation software be used to evaluate motion recordings for other application areas such as gait analysis or medical science?

In many scenarios, animation software provides tools for visualisation, kinematic models and other features that are important to motion capture. However, it is difficult to repurpose animation software for experimental research. The following list of software is compatible with motion capture file formats. However, it highlights a need for research-oriented inertial motion capture software environments.

2.9.1 Autodesk 3ds Max

3ds Max [132] is the industry standard for modelling, rendering and animation. It provides users with a vast array of tools for mesh sculpting, texturing and animating 3D models. Its latest iterations feature Character Studio, a plugin facilitating more advanced character animations. Character Studio can be used to animate mesh geometries through custom bone rigs. However, the user is forced to implement a prefabricated skeletal armature called *biped* when working with motion capture data. The highly customisable biped is the quickest technique for visualising BVH motion data.

2.9.2 MotionBuilder

MotionBuilder [133] the most advanced 3D animation software application available commercially and provides full support for motion capture file formats such as BVH, BVA and ASF/AMC. Interoperability with other software environments is achievable through native FBX support. MotionBuilder's SDK can be used to develop plugins that extract data from inertial motion capture suits. In the context of inertial motion capture, MotionBuilder's physics engine can be used to simulate horizontal displacement by applying gravitational forces and collision detection to skeletons. Even though MotionBuilder does provide an SDK, it is difficult or impossible to modify any fundamental aspects of this software application (e.g. the kinematics).

2.9.3 Maya

Maya [134] is a very popular software environment for modelling, rendering and animation. Its application areas target the filmmaking and game development. Natively, Maya does not support any motion capture formats. However, the BVHImportExport [135] plugin can be installed to provide compatibility for BVH files formats. Additionally, it is possible to load ASF/AMC files into Maya by first altering the motion data using the *AMC2MOV* and *ASF2MEL* converters [136]. To summarise, it is possible but difficult to use Maya for visualising motion capture data.

2.9.4 Blender

Blender [137] is an open source modelling, rendering and animation suite developed by the Blender foundation. A large community of professionals and enthusiasts support Blender by developing plugins. Blender's character animation tools support bone rigging and key frame animation. This software application does contain BVH importer can be used to map motion capture data to skeletons. Blender's gaming engine shows potential for real-time motion capture applications.

2.9.5 Other Software Applications

There are several smaller and less known applications aimed at motion capture animation. Life Forms Studio [138] and Poser [139] are two character animation packages that support BVH and other motion capture file formats. Carrara [140] is a cost-effective software environment developed by Daz3D. Its latest iterations provide functionality for importing, editing and exporting BVH files.

2.9.6 Comparative Evaluation of Software Applications

This section summarises the benefits of using each of the aforementioned software applications by creating a summary of the advantages and limitations of each solution. This study identifies BVH as being the most popular file format and outlines two fundamental problems. First, there are no software applications designed primarily for motion capture. Second, there are no software applications that facilitate experimental research in the field of inertial motion capture and BSNs. Table 2-2 illustrates a comparison of the software applications and summarises three key aspects: advantages that are relevant to motion capture, supported file formats and how hardware integration is possible.

Table 2-2: Comparison of software applications in the context of motion capture.

Software	Advantages	File Formats	Hardware Integration
3ds Max	- kinematic character rigs - animation tools - biped primitive - popularity	- BVH	- through plugins
MotionBuilder	- kinematic character rigs - powerful animation tools - physics engine - good file format support - popularity	- BVH - BVA - ASF - AMC	- using the SDK - through plugins
Maya	- kinematic character rigs - animation tools - popularity	- BVH	- through plugins
Blender	- kinematic character rigs - animation tools - open source - free - community-driven	- BVH	- through plugins - source code
Carrara	- kinematic character rigs - animation tools	- BVH	- N/A
Life Forms Studio	- kinematic character rigs - animation tools - good file format support - aimed at motion capture	- HTR - BVA - BVH - LWS - DXF - ACC	- N/A
Poser	- kinematic character rigs - animation tools - good file format support - aimed at motion capture	- BVH - PZ2	- N/A

2.10 Conclusions

This chapter has introduced and examined motion capture as the topic of research for this thesis by looking at application areas, hardware technologies, inertial BSNs, inertial sensors, the principles behind kinematics, motion capture file formats and software. A set of problem areas is concluded as the result of this literature survey.

This chapter begins by defining the concept of motion capture as the process of using sensors and software to reconstruct a virtual representation of real-life movement. While the principles behind motion capture are not new, the ability to design and implement motion capture systems is only recently possible due to the miniaturisation of MEMS and the availability of superior computational resources. Motion capture is chosen as the primary topic of research because it is relevant to a wide range of application areas while it proves problematic and challenging in many aspects, particularly in terms of usability. A better motion capture workflow could simplify the interaction between users and their system to make inertial BSNs more ubiquitous.

This chapter continues to analyse what motion capture is used for by identifying research projects, as found in literature, and grouping them into application areas. The five main application areas are identified as being: animation, real-time motion capture, medical science, gait analysis, ergonomics and robotics. However, the lines between these application areas are blurred. For example, gait analysis may be considered a medical science or sport science application. Some application areas are result-driven while others are accuracy-driven. A medical science experiment will demand accuracy while video games may require a visually interesting animation.

Once application areas are identified, this chapter discusses hardware technologies and their impact on the field of research by looking at major industry contributors and what systems are available commercially. While there are many approaches for capturing motion, there are only three main technologies: optical (e.g. Vicon, Optotrak, Microsoft Kinect), inertial (e.g. XSens, Animazoo), mechanical systems (e.g. eMove, Gypsy 7). Optical, mechanical and inertial motion capture technologies present strengths and weaknesses. Motion capture hardware development is a result-driven process where systems are tailored to specific application areas. Optical systems require expensive recording studios and limit the motion recording process to specific environments due to occlusion and lighting problems. Despite technological advances that have reduced the cost of sensors, inertial motion capture systems remain expensive. Mechanical systems are cumbersome and lack the accuracy required for specific tasks. This chapter concludes a comparative evaluation of all motion capture systems and mediums in terms of advantages, disadvantages, application areas and cost. Based on these factors, inertial motion capture is chosen as the primary research topic for this thesis.

The first step in understanding inertial technologies is to understand how IMUs function. Emphasis is put on the three constituent sensors contained in an IMU, namely the gyroscope, accelerometer and magnetometer. The chapter continues to discuss what attributes describe the performance of a sensor in terms of: number of sensing axis, full scale range, sensitivity, bandwidth, shock tolerance, drift, noise, size, power consumption and output formats. Sensor fusion is discussed to create an understanding of how data originating from three different sensors can be merged to produce a more accurate reading.

Having defined the fundamentals of IMUs, this chapter discusses how several devices can be interconnected to form a BSN. In the context of inertial motion capture, BSNs are synonymous with animation suits. BSNs can be defined as being directional or omnidirectional, homogenous or heterogeneous, software or hardware-centric. Each type of BSN has distinct advantages and disadvantages.

Once data is obtained from hardware, kinematic models are required to reconstruct the motion in 3D. The concept of a kinematic model is introduced as a skeletal rig used to mimic the original motion in virtual space. For inertial motion capture systems, kinematic models are vital in computing dead reckoning. Because inertial systems contain no sensors that are dedicated to measure horizontal displacement, kinematic modes are required to synthesize an approximation of dead reckoning through planar collision detection and foot placement estimation.

This chapter finishes with an overview of motion capture software applications and file formats to create an understanding of how sensor hardware can be used in conjunction with a computer. As demonstrated by the comparative evaluation, inertial motion capture systems are regarded as new technologies and have little support from computer software applications. The software applications that do support motion capture are primarily designed for animation purposes and provide little to no functionality for integrating and researching hardware. Therefore, it is up to hardware developers to create software environments or plugins. For that reason, there is no standardised workflow for utilising inertial motion capture systems. Consequently, this thesis proposes a new workflow through a novel inertial motion capture framework.

2.10.1 Problem Areas

Aside from introducing the principles behind motion capture, the main purpose of this chapter has been to identify general research trends and problem areas that will benefit from further research and development. This section discusses three specific problems are that were influential in concluding this thesis' problem statement (see

Section 1.2). This list of problems is extended with further research found throughout this thesis.

Software Environments

The development of motion capture systems requires a software environment. Motion capture data is unusable unless processed and applied to a kinematic model. Researchers are producing prototype software applications that are tailored to their experiments, which is a time-consuming process. Each system works in a slightly different manner and there are no standards for developing BSNs. A research-grade motion capture software environment, implementing kinematics and providing tools for system integration, could be used as a baseline for establishing an effective and more standardised workflow. As previously discussed, there are no existing software environments aimed specifically at motion capture. The only solutions that exist are animation software packages, which provide tools for visualisation, kinematic models and other features that are important to motion capture. However, motion capture is complex it is difficult to repurpose animation software for motion capture purposes (such as experimental research).

Body Sensor Networks

Inertial motion capture systems are primarily software-centric whereby data is gathered from the sensors at high frame rates and communicated to the computer frequently. This is because data cleaning and sensor fusion algorithms compress larger amounts of data to produce accurate animations. As a result, inertial motion capture suits are overcomplicated, fragile and cumbersome. By making suits more hardware-centric, it is possible to simplify hardware and produce more robust, versatile and modular systems. This concept is further discussed and demonstrated in Chapter 4 by the Motion Tracking Development System (MTDS) prototype.

File Formats and Standards

Motion data can be stored, transferred and accessed through several file formats. A study presented in this chapter revealed that motion capture file formats are mainly supported by animation software applications. Other application areas will render these formats incomplete and inefficient. As demonstrated in the next chapters, an extended version of BVH, or another format, could standardise motion capture file formats if extended to meet the requirements of a modern motion capture system. This thesis also proposes that web technologies could be used to store and process motion data, thus making file formats redundant by replacing local data storage with online data storage.

Dead Reckoning

Inertial motion capture's main downfall is dead reckoning, a popular research topic focused on foot placement estimation. Unlike optical motion tracking devices, inertial suits do not have the ability to measure the performer's exact position within the environment. Several solutions, such as the lowest-point algorithm, produce a rough estimation of the user's world-space location. That location is not sufficiently accurate because of calibration errors and anchor selection problems. The lowest-point algorithm could benefit from the further computations involving the physical properties of the motion performer. This problem led to the development of the lowest-anchor centre of mass algorithm (LACOMA) presented in Chapter 3.

CHAPTER THREE

3 Inertial Motion Capture Software Environments

3.1 Introduction

The preceding chapter has introduced motion capture, analysed several technologies while focusing on inertial sensing and discussed many application areas. The preceding chapter has highlighted the importance of this research topic while revealing problem areas and limitations. The reason behind its recent advances is the affluent computer graphics industry that relies on time-efficient and cost-effective techniques for achieving realistic character animations. Inertial motion capture trails behind its optical rival and presents few standardisations in terms of usability workflows and file formats. While there are numerous tools for animation editing, there are no bespoke software solutions that provide suitable experimentation environments for motion capture research. For this reason, researchers are wasting time repurposing animation frameworks for use in behavioural studies.

This chapter presents the Skeletrix software environment, which is designed to encourage new projects in the field of inertial motion capture by providing users with a platform for developing body sensor networks (BSN). Its first purpose is to provide a suitable experimentation environment, accompanied by programming scaffolding and a driver development kit, for users interested in studying or engineering inertial measurement units (IMU) that enclose gyroscopes, accelerometers and magnetometers. Its second purpose is to support the research showcased in the following chapters. Notably, this software environment solution is not presented as a motion-editing tool for character animation and does not replicate the core functionality found in other software.

The term software environment is often considered ambiguous and coincides with the term framework. This thesis refers to inertial motion capture, as a whole, as a framework. To operate, motion capture frameworks require the development of electronics, microcontroller firmware and software environments. The software environment element denotes a multifaceted sequence of procedures for gathering, refining, visualising and storing motion data. As shown in Figure 3-1, each procedure envelops a wide range of computer science domains ranging from interactive visualisation techniques to data intercommunication protocols.

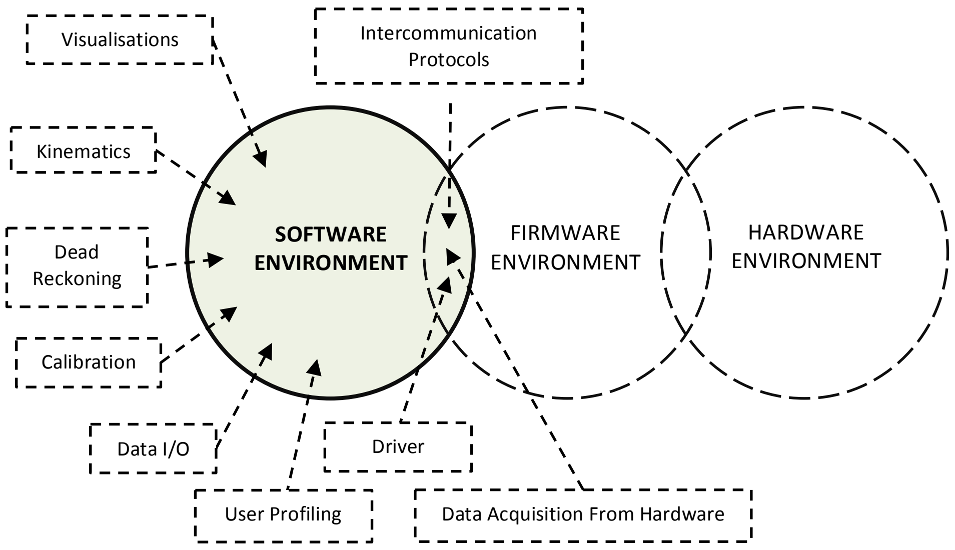


Figure 3-1: A high-level representation of motion capture frameworks with emphasis on the software environment aspect and its technologies.

This chapter presents the development lifecycle of the Skeletrix software environment with emphasis on three unique aspects: DDK, LACOMA and BVHE.

The driver development kit (DDK) is developed to improve the communication mechanism between software applications and hardware through a modular approach that can be customized to suit the needs of any inertial motion capture system. The DDK is then used in the following chapter, Chapter 4, to integrate and develop new hardware within the framework.

In response to the file format standardisation problem, this chapter introduces Biovision Hierarchy Extended (BVHE), a new file format combining hardware configuration properties with inertial data, thus improving the relationship between software and hardware.

Dead reckoning remains a major limitation for inertial motion capture systems. This chapter proposes the lowest-anchor centre of mass algorithm (LACOMA) Pascu et al. [18], which utilizes the musculoskeletal centre of the human body to compute the support foot during gait more accurately. The proposed solution is based on the principles of the lowest-point algorithm [127].

3.2 Framework Relevance

The software environment presented in this chapter is a major component of the Skeletrix framework, which provides a preliminary understanding of inertial motion capture in terms of extracting data from hardware, processing motion, reconstructing kinematic motion and rendering 3D visualisations. The Skeletrix software environment facilitates the integration of existing BSNs that are available commercially and the development of new BSNs. For example, the software environment was used to develop the Motion Tracking Development System (MTDS) prototype, an upper body motion capture suit presented in Chapter 4. Its source code was also used as a starting point for the smartphone-driven BSN presented in Chapter 5. Figure 3-2 highlights the software environment's position within the framework as a software layer between inertial hardware and the Motion Cloud, which is presented in Chapter 6.

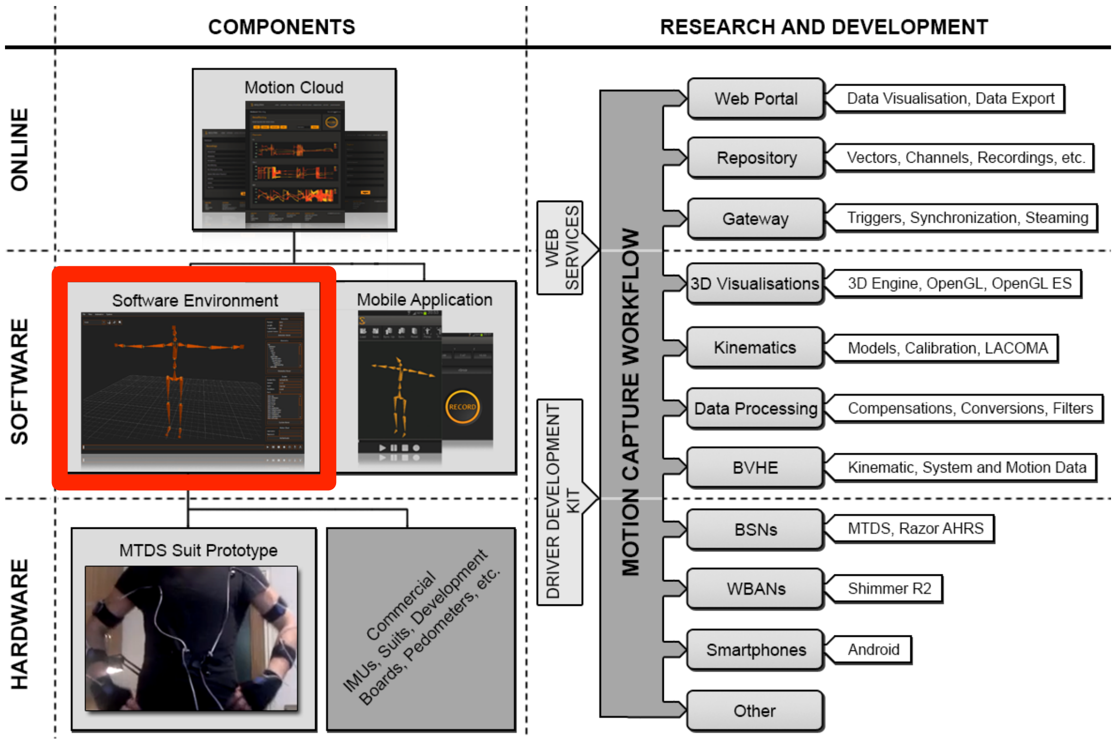


Figure 3-2: The Skeletrix software environment provides an invaluable set of tools for extracting inertial data, constructing BSNs and is an important component of the Skeletrix framework. **Source: Pascu et. [15] [16]**

3.3 Preliminary Requirements Specification

The development of the Skeletrix software environment combines a wide range of software development domains that require careful planning.

3.3.1 System Requirements

The first step, which precedes any design phase, involves identifying all the relevant software development paradigms. The methodologies adopted in its conception must adhere to industry-wide programming standards.

Transparency: Most proprietary motion capture software environments are exclusive to pre-specified technologies. In most cases, functional particularities are hidden from the user in order to protect intellectual property. For this reason, most developers in this field have adopted the black box approach to software development that blinds users from viewing, understanding and thus customizing the system's functionality. The Skeletrix software environment must be designed transparently to ease the user's task of integrating new motion sensing devices. In this context, transparency refers to a superior level of access where the user can observe and learn the software environment's inner workings through its interface. As shown in Figure 3-3, transparency follows the white box model whereby the software environment's inner workings are made visible to the user.

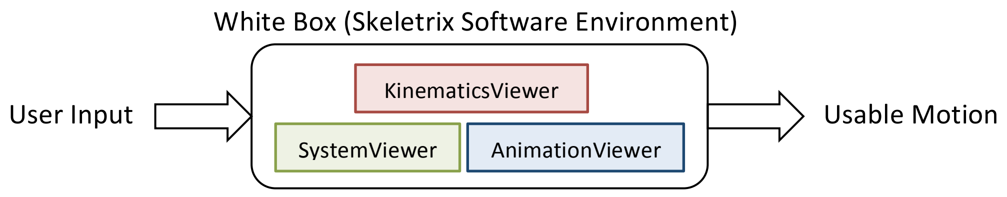


Figure 3-3: The white box model applied to the Skeletrix software environment.

Object-Orientation: It is good practice to logically distribute code into components, each having a distinct purpose. According to code recycle theory, each component must have a generalized purpose so that it may be reused beyond its initial implementation. According to encapsulation theory, each component must limit incoming communications to a set of publicly accessible functions or sockets. Encapsulation is important in achieving efficient object intercommunications. The Skeletrix software environment presented in this chapter must address these programming principles to succeed.

3.3.2 Functional Requirements

Figure 3-4 introduces the Unified Modelling Language (UML) in the design flow to establish a list of functional requirements. UML's use case diagrams are valuable in creating a visual understanding of a framework's features while defining user interactions. The following diagram depicts a list of possible user-performed actions. Subsequently, each action is elaborated to conclude an implementation approach.

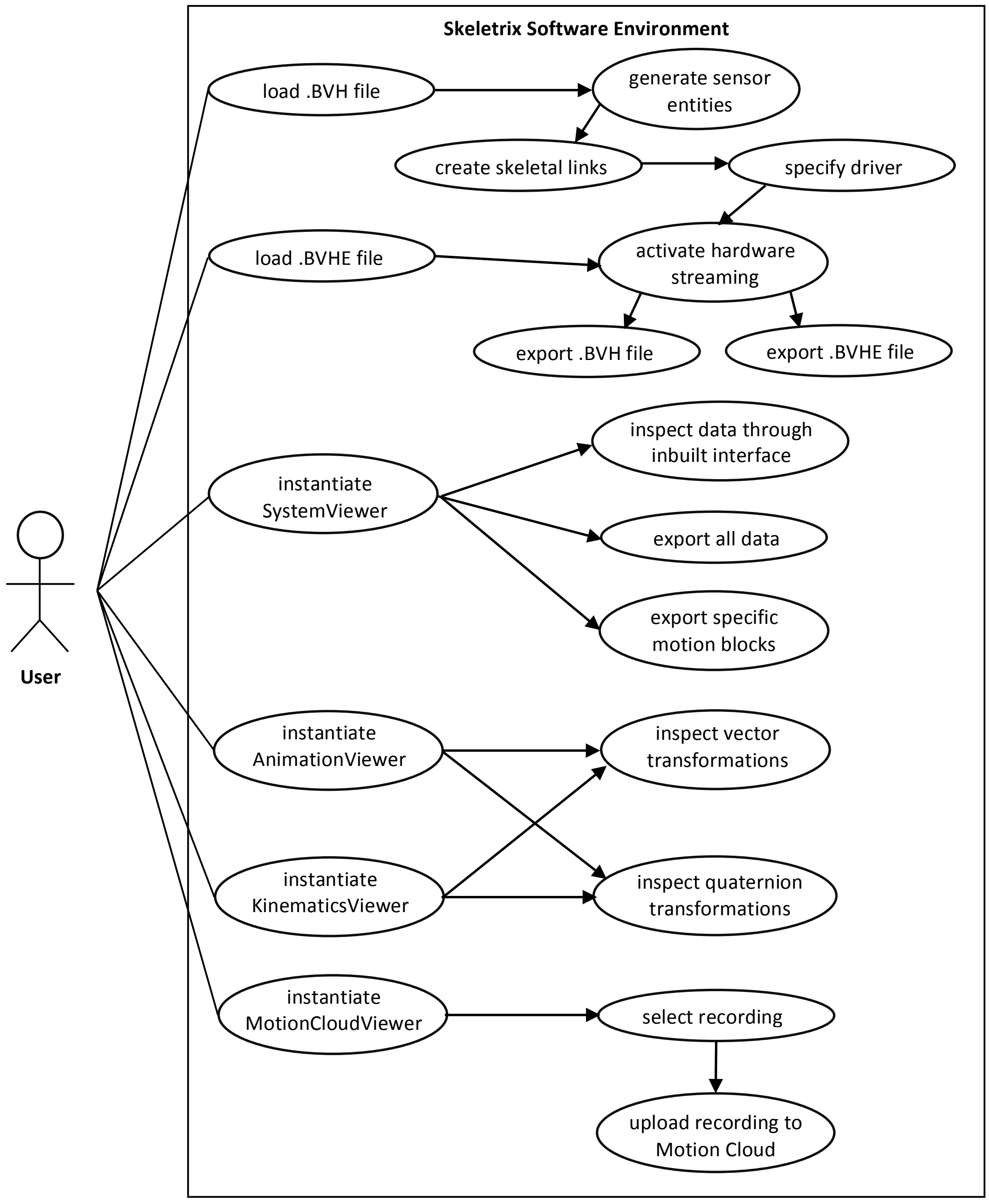

Skeletrix Software Environment
load .BVH file
generate sensor entities
create skeletal links
specify driver
load .BVHE file
activate hardware streaming
export .BVH file
export .BVHE file
instantiate SystemViewer
inspect data through inbuilt interface
export all data
export specific motion blocks
User
instantiate AnimationViewer
inspect vector transformations
instantiate KinematicsViewer
inspect quaternion transformations
instantiate MotionCloudViewer
select recording
upload recording to Motion Cloud

Figure 3-4: Use case diagram highlighting the user performed actions in the Skeletrix software environment.

Data Import: As shown in Figure 3.4, the user must be able to import motion data using either the Biovision Hierarchy (BVH) or BVHE file type. This requirement is important to simulating a system's behaviour in the absence of hardware. Once the user has chosen the relevant file by using a file dialog, the Skeletrix software environment must convert its syntax into usable data structures. Because BVHE code contains BVH syntax, a shared lexical analyser can be developed to parse both file types. Finally, if the user choses to import a BVH file, its BVHE supplements should be generated automatically in accordance to a set of rules. Those rules are subsequently discussed in the implementation sections of this chapter.

Data Export: Once a motion capture recording session has completed, the user may wish to store data in BVH or BVHE files. BVH must be supported for cross-compatibility with existing software environments. BVHE is required in situations where the user choses to save the session's system data for future use (e.g. recording the motion of an individual multiple times without having to reconfigure the software environment). In both scenarios, this requirement implies the development of a compiler that can generate valid BVH/BVHE syntax. In some scenarios, the user may choose to inspect the motion data by plotting graphs or generating mathematical statistics. In those circumstances, data of interested should be exportable in tabular formats that are suitable for scientific analysis.

Driver Interconnectivity: The user must be able to choose a specific dynamic-link library (DLL) driver to communicate with motion capture hardware. DLLs are an efficient and highly accessible approach for encapsulating code that has one distinct purpose: extracting data from the hardware. The Skeletrix DDK will provide a set of instructions and a template allowing third party users to develop such drivers. Depending on the computer in use, a driver will always require the specification of a communication port. The communication port value should be detected automatically or inputted manually by the user.

Calibration: The user must be able to calibrate a singular inertial sensor or a more complex BSN through the standard T-Pose technique. This technique works by asking the motion performer to remain motionless with both arms extended laterally away from the body while the software environment compensates the rotational difference between the person's posture and the kinematic model.

Viewer System: In this context, viewers are categories of tools for examining specific software properties. The user must be able to inspect the software's motion data in real-time. In a motion capture simulation platform, motion data is denoted by positional and rotational transformation sequences. A transformation can be examined by reading its numeric values in a comprehensible format. The software environment contains three components of interest to the user: the kinematic model, the system model and the animation model. A viewer must accompany each model and every viewer must be given an additional user interface allowing each data entry to be analysed and rendered independently. This requirement is in accordance with the principle of transparency introduced in the previous section.

3.3.3 Interface Requirements

The interface development process is crucial in achieving a rigid relationship between the user and the software environment. The features previously mentioned need to be easily accessible and more importantly, intuitive. The Skeletrix software environment must feature an interface that contains both 3D rendering and other 2D elements.

HCI Principles: The Skeletrix software environment must adhere to all Human Computer Interaction (HCI) conventions whereby each interface component is recognisable, easily accessible and annotated appropriately. The overall interface must be simplified to facilitate a time-efficient user experience that requires little familiarization. The steps taken by a user in performing a task must be kept to a minimum by optimizing the number of interface elements. The interface must provide a level of consistency whereby all interface objects are designed and built in accordance to a set of predetermined rules.

3D Motion Reconstruction: Due to the three-dimensional nature of character animation, the software environment's interface must implement a 3D engine and a renderer to display kinematic models. The visualization of a kinematic model implies drawing skeletal hierarchies of bones where each bone is depicted by a 3D topology.

Viewport Interaction: Defining this task's requirements can be challenging as computers are natively designed to work with two-dimensional input devices, such as mice and keyboards. The use of 2D input devices in 3D space requires dimensional simplification. Each element constituting a 3D interaction can be broken down into its degrees of freedom (DOF). For example, the rotation of a camera can be simplified to singular degrees of freedom corresponding to yaw, pitch and roll. Chapter 2 conducted an overview of existing software environments that are relevant to the field of motion capture. An interaction technique, which requires little familiarization from potential users, should be chosen by studying how interactive 3D is achieved throughout industry-acclaimed products (e.g. those previously introduced in Section 2.9).

Multimodal Visualizations: Visualizing a skeletal rig from multiple angles is helpful in creating a more thorough understanding of individual bone behaviours. A predetermined set of camera configurations will allow users to toggle between view angles (i.e. lateral, frontal, perspective, orthographic, etc.) without having to navigate the viewport manually. Alternatively, several viewports could be rendered concurrently to visualise the skeletal rig from multiple angles simultaneously. Researchers focusing on person dead reckoning may be interested in observing the rig's interaction, in terms of horizontal displacement, with its surroundings. As a result, the entire interface should be developed to increase the size of the window so that it fits the screen dimensions.

3.4 Architecture Overview

The Skeletrix software environment architecture follows the principles of object-oriented programming. The code is distributed throughout C++ classes while relying on the QT [141] for visuals and interfaces. QT was chosen because of it provides a great set of tools for developing user interfaces that integrate OpenGL. Figure 3-5 denotes the major software elements. The architecture can be divided between a back-end layer and an interface layer. The back-end layer consists of data models (red), mathematics libraries (green) and data I/O (blue). Concurrency and software timers are also a key component in ensuring that the software environment functions efficiently. The interface layer encompasses: the main GUI (red), the viewers (green) and the 3D renderer (blue). All these elements are discussed throughout this chapter in more detail.

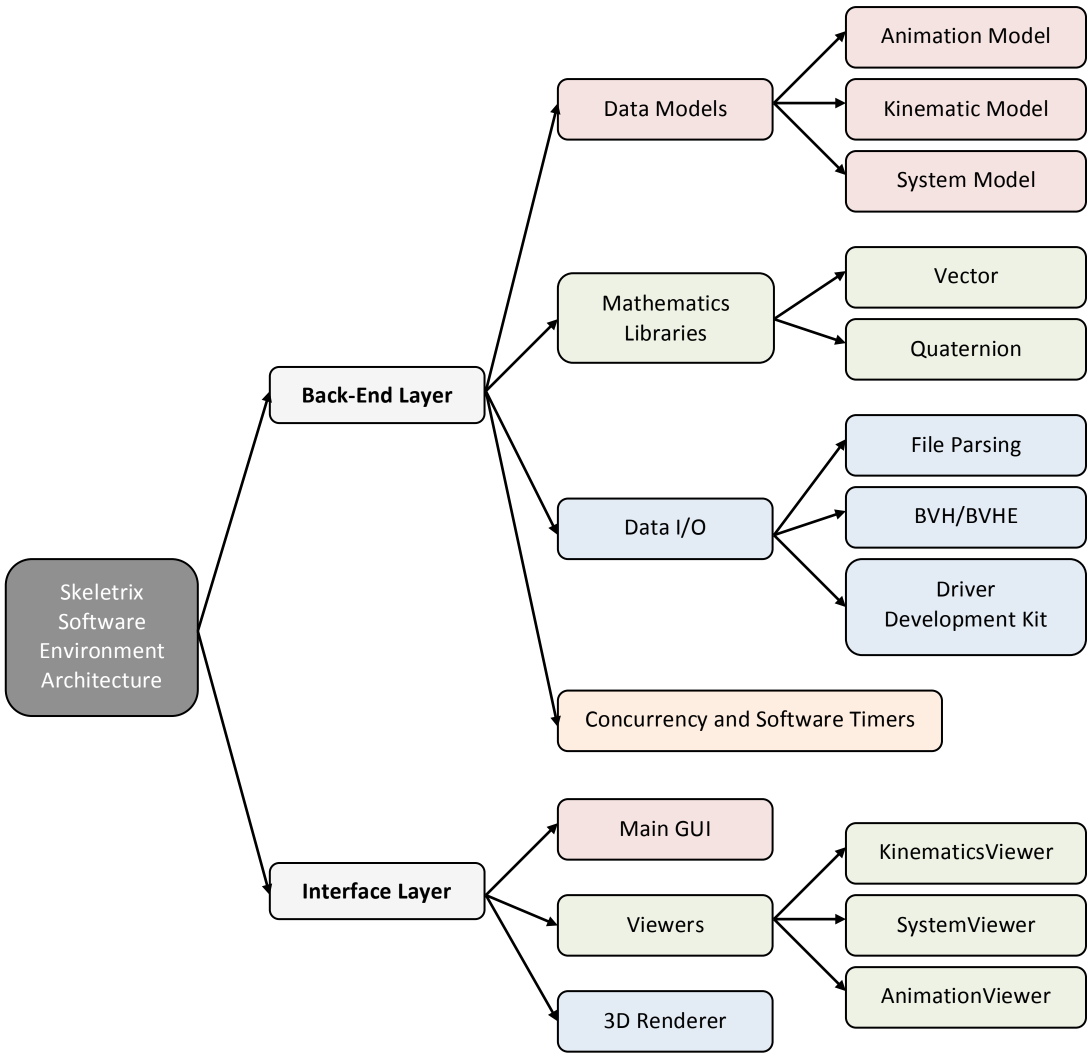


Figure 3-5: Skeletrix software environment architecture denoting the components that form the back-end and interface layers.

3.5 Data Models

As previously discussed, the software environment holds data at within three different models: the animation model, the kinematic model and the system model. The animation model stores rotational data, the kinematic model stores skeletal data and the system model stores system-specific configuration data.

3.5.1 Animation Model

The animation model is an integral component of the software environment, situated between the kinematic model and the system model. In accordance to the code reuse/recycle theory, this component is designed as an interchangeable module with potential uses outside the scope of the software environment (e.g. in the mobile application presented in Chapter 5). Its first purpose is to store all incoming hardware-related data during recording sessions. Its second purpose is to feed the kinematic model (and subsequently the 3D engine) the rotational transformations necessary during animation playback.

In the playback scenario where the user is viewing pre-recorded data, the animation model closes all incoming communication channels. While constantly listening for user action, it provides all the functionality for playing, pausing and stopping the animation. Once the user triggers animation playback by pressing the play button, the animation manager begins to iterate through its data structures, constructs a packet for each frame of motion and sends that packet to the kinematic model and subsequently to the renderer.

In the recording scenario, the animation manager opens all incoming communication channels and listens for system model messages. To visualize the motion as it is performed, the animation manager must bypass the outgoing playback signal and perform two tasks concurrently by employing two threads. The first task is to gather and store all incoming rotations while ensuring that no data is lost. The second task is to echo all incoming rotations straight to the kinematics model. Without concurrency, the functionality of real time motion visualization would be unachievable.

As previously mentioned, the majority of inertial motion data consists of positional vectors and rotations. In the Skeletrix software environment, rotations are stored as quaternion data structures, which are significantly faster to compute than the Euler equivalent. There are numerous methodologies for storing this type of data (e.g. simple floating-point precision numbers arrays). To optimize intercommunications, the animation model's design takes into consideration the external needs of both system and kinematics models. Unique identifiers are used inform the software environment of each reading's timestamp (the animation frame value) and destination (the skeletal object to which it is applied).

Motion data is organized as a collection of tape objects. Tape objects are linear arrays enclosing the complete information required to reconstruct one frame of motion. Specifically, the first tape delineates the animation's template by declaring one spatial positioning channel and multiple bone identifiers. In accordance to this template, each frame of motion will begin with a positional vector (used to displace the skeletal rig's root) followed by a list of quaternion rotations (one for each bone). As a result, the animation model's outgoing package must contain both the template tape and a frame tape. With that information, the kinematic model will proceed to assign transformations to the skeleton. Figure 3-6 illustrates this data structure in more detail.

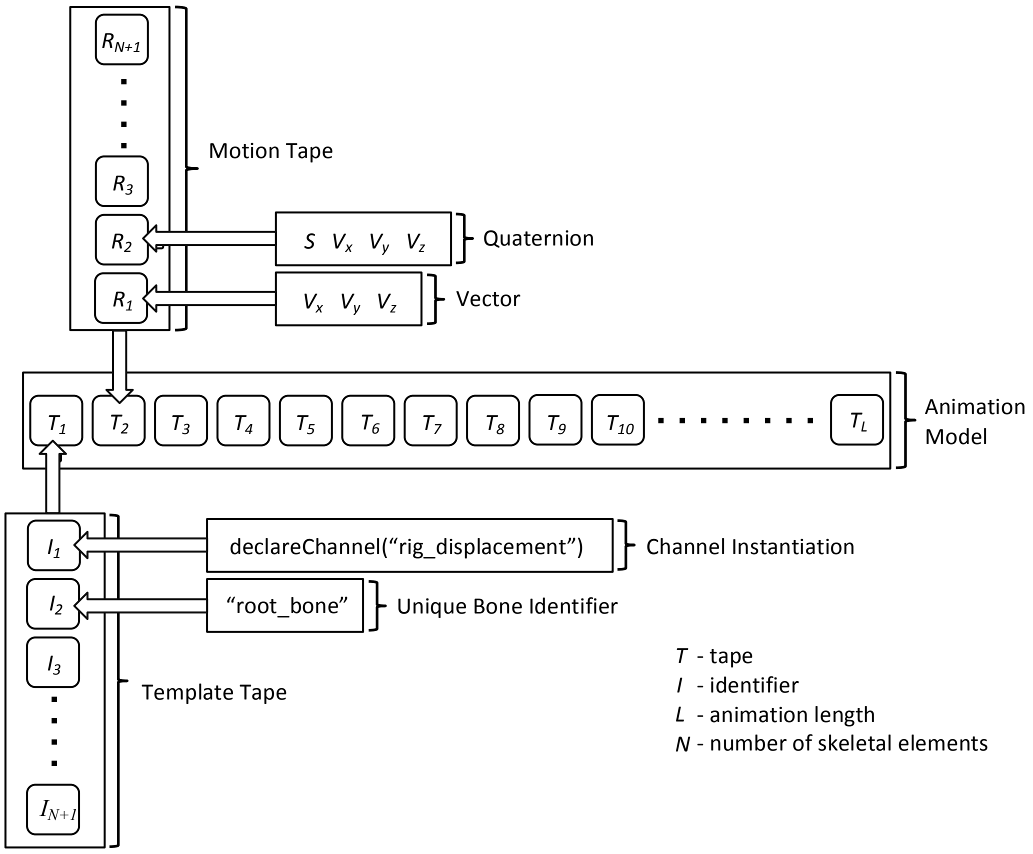


Figure 3-6: Animation model data structure enclosing one template tape and a list of motion tapes.

3.5.2 Kinematic Model

The kinematic component of the Skeletrix software environment is used to both visualize and process the incoming motion data to construct a skeletal rig. It is formed of a universal skeleton object that encloses a list of bones. To reconstruct a frame of motion, the skeleton retrieves data from the animation component and applies it recursively to that list of bones. The contained hierarchy cannot be traversed linearly because of the nature of inverse kinematics where transformations must be applied in a very specific order. In accordance to the specification of BVHE, the skeleton needs to support several types of data depending on the system in use. The incoming stream of angular readings, from the system object, may enclose local or world-space rotations depending on the system in use. In either case, the kinematic model will convert those rotations into local-space rotations. To support several types of information, each bone object is defined by four quaternions: local, world, correction and calibrated. The local quaternion represents a local-space rotation, as required for the animation model, BVH or BVHE files. The world quaternion represents a world-space rotation, as required by the 3D engine. The correction and calibration quaternions store angular compensations.

3.5.3 System Model

The system model represents an intermediate stage of computations between the hardware and the kinematic model. The system and kinematic models share the same design with one fundamental difference: there is no object hierarchy. This is because daisy chained sensors do not require an object hierarchy and all skeletal information, which interconnects bone objects, is stored in the kinematic model.

The system model contains sensor objects (virtual IMU objects), which are stored in a linear array. Those objects create a virtual representation of the BSN connected to the software environment. Therefore, the system model stores incoming rotations only temporarily before the data is applied to the kinematic model.

Each sensor contains four rotational channels: *gyroscope*, *accelerometer*, *magnetometer* and *fused*. Provided that the system does not compute firmware-side sensor fusion, the first three channels are utilized for storing incoming data corresponding to gyroscopes, accelerometers and magnetometers readings. The *fused* channel will hold the processed motion if sensor fusion algorithms or filters are used.

One additional channel could be included to store positions for optical systems. Even though the software environment is not fundamentally designed to function with optical data, optical system could be integrated in the future.

3.6 Interface Layer

This section illustrates visually how the Skeletrix software environment operates and outlines the user experience. It discusses the four principal interface layer objects: core GUI, KinematicsViewer, SystemViewer and AnimationViewer. As previously mentioned, the design of the interface is crucial in achieving an intuitive user experience. The Skeletrix interface layer, designed and developed using QT [141], presents a combination of 2D and 3D elements for visualizing and interacting with motion capture data. Much attention was given to streamlining the user experience by simplifying the number of visual elements while maintaining the technical functionality specified in the requirements analysis.

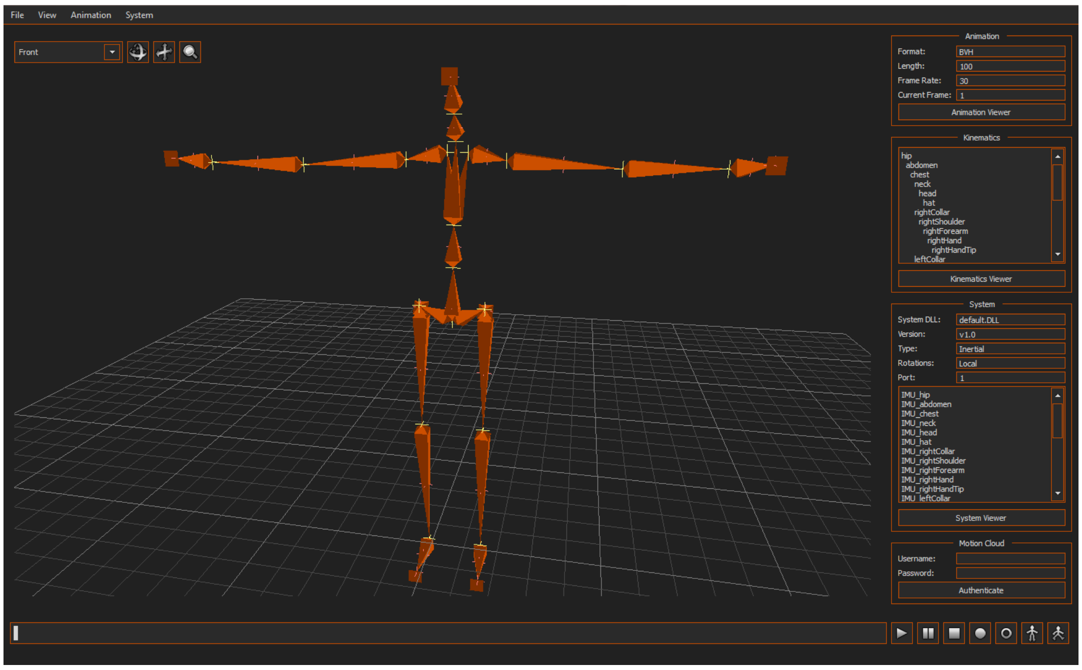
The communication between the software environment's interface layer and back-end architecture is omnidirectional and achieved through signals and slots [142]. Each interface element transmits a signal in response to a user-performed action. QT provides a list of events that can be used to trigger signals (i.e. mouse click, mouse drag, mouse hover, keyboard input, etc.). In this context, a slot is a special type of function that is accessible by the interface.

Although irrelevant to the field of research, the software environment's aesthetics are important in the computer graphics industry. Competing developers are designing interfaces that deliver both functionality and pleasant visuals. QT relies on cascading style sheets (CSS) to format the look and feel of each interface element. Therefore, the development of the software interface is similar to that of a website, making the interface highly customizable. To restyle the Skeletrix software environment, users are given access to an external CSS file enclosing the default template code.

The interface also aims to provide users with a detailed understanding of the software environment's inner workings. All back-end transformation sequences, used to convert raw motion into usable data, are on display. At the kinematic level, the interface displays the stages of calibration (previously discussed in Section 2.5.2) through which angular compensations, representing the rotational difference between the kinematic rig and the motion performer, are calculated. At the system level, the interface displays the data obtained from the hardware. Once all the data is processed, the final result is displayed in the AnimationViewer interface.

3.6.1 Core GUI

The core GUI shown in Figure 3-7, is the largest interface layer component. It comprises of four logical subdivisions: a multimodal canvas, a set of drop-down menus, an information toolbar and a slider for animation playback. To begin with, the software's back-end will be launching a thread to instantiate and render the viewport. The viewport camera can be repositioned and rotated manually or by selecting a pre-set. The pre-sets allow users to view the animation from the top, left, front and perspective. Situated in the header, the dropdown menus provide the customary functionality (i.e. importing and exporting motion files). Situated vertically on the right hand side, the information toolbar contains three panes that exhibit a simplified overview of the animation data, kinematic skeleton and hardware configuration. Each pane can launch a viewer object for a more elaborate visualization of the information. Situated in the footer, the slider displays the animation timestamp along with the conventional controls for playing, pausing, stopping and recording the animation.



Format: BVH
Length: 100
Frame Rate: 30
Current Frame: 1

hip
abdomen
chest
neck
head
hat
rightCollar
rightShoulder
rightForearm
rightHand
rightHandTip
leftCollar

System DLL: default.DLL
Version: v1.0
Type: Inertial
Rotations: Local
Port: 1

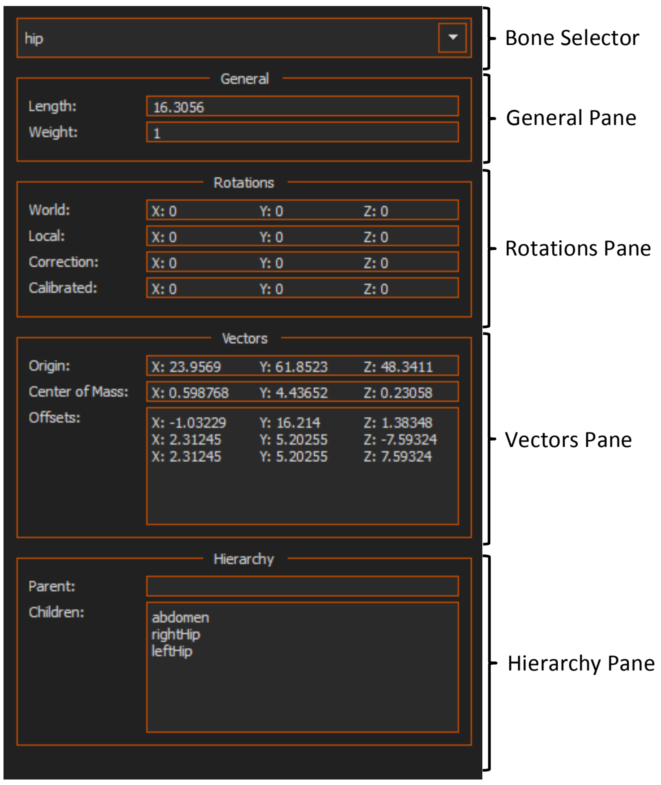
IMU_hip
IMU_abdomen
IMU_chest
IMU_neck
IMU_head
IMU_hat
IMU_rightCollar
IMU_rightShoulder
IMU_rightForearm
IMU_rightHand
IMU_rightHandTip
IMU_leftCollar

Username:
Password:

Figure 3-7: Core GUI illustrating the landing of a pre-recorded jump animation.

3.6.2 KinematicsViewer

As shown in Figure 3-8, the first of the three viewers, the KinematicsViewer, allows users to view all the vector positions, quaternion rotations (which are converted into Euler rotations by the interface for readability reasons) and kinematic model information. Once a motion file is imported, Skeletrix will instantiate this interface and populate it with objects. Users can select individual skeletal objects for inspection. As a bone is selected, data is retrieved from the back-end and displayed within four panes: general, rotations, vectors and hierarchy. The general pane displays the calculated lengths and weights of bones as floating-point numeric values. Weight models are introduced in the next sections in the context of person dead reckoning and the LACOMA algorithm. The rotations pane shows the local space, world space, corrected and calibrated transformations converted from quaternions to Euler for readability reasons. The vectors pane displays the origin, centre of mass and lists several offsets as bones may have multiple end points. The hierarchy pane encloses the title of the parent bone and lists the children. By inspecting these panes, the user will gain a better understanding of how the kinematic model is computed and identify any problem areas.


hip

General

Length: 16.3056
Weight: 1

Rotations

```
World:       X: 0    Y: 0    Z: 0
Local:       X: 0    Y: 0    Z: 0
Correction:  X: 0    Y: 0    Z: 0
Calibrated:  X: 0    Y: 0    Z: 0
```

Vectors

```
Origin:          X: 23.9569     Y: 61.8523    Z: 48.3411
Center of Mass:  X: 0.598768    Y: 4.43652    Z: 0.23058
Offsets:         X: -1.03229    Y: 16.214     Z: 1.38348
                 X: 2.31245     Y: 5.20255    Z: -7.59324
                 X: 2.31245     Y: 5.20255    Z: 7.59324
```

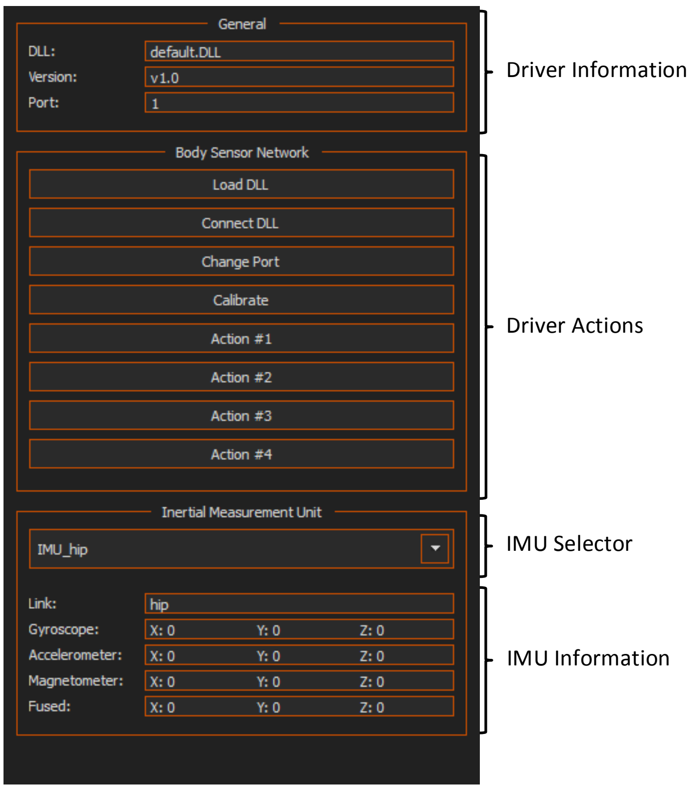
Hierarchy

Parent:
Children:
abdomen
rightHip
leftHip

Figure 3-8: KinematicsViewer illustrating the information of a hip bone.

3.6.3 SystemViewer

As shown in Figure 3-9, the second of the three viewers, the SystemViewer, allows users to analyse the incoming stream of data as obtained from the hardware. In many respects, this interface resembles the KinematicsViewer. The SystemViewer has two distinct sections for configuring hardware and for inspecting individual sensors. To begin with, the hardware section displays the overall system data as found in BVHE files. That data is used to specify the driver, communication ports and other general properties. Several buttons allow the user to modify the driver configuration and perform system actions such as T-Pose calibration. Four additional actions are provided for systems that require specific handshaking instructions (e.g. the Animazoo IGS suits require Northing). Afterwards, the user may choose to inspect individual sensors. Sensor attributes are distributed throughout three panes that show general driver information, driver actions and IMU information. Once an IMU is selected, the rotations pane will display incoming sensor readings from gyroscopes, accelerometers and magnetometers. If sensor fusion is implemented, the fourth channel stores the fused result (i.e. the combined and filtered motion of the gyroscope, accelerometer and magnetometer).


General

DLL: default.DLL

Version: v1.0

Port: 1

Body Sensor Network

Inertial Measurement Unit

IMU_hip

Link: hip

Gyroscope: X: 0 Y: 0 Z: 0

Accelerometer: X: 0 Y: 0 Z: 0

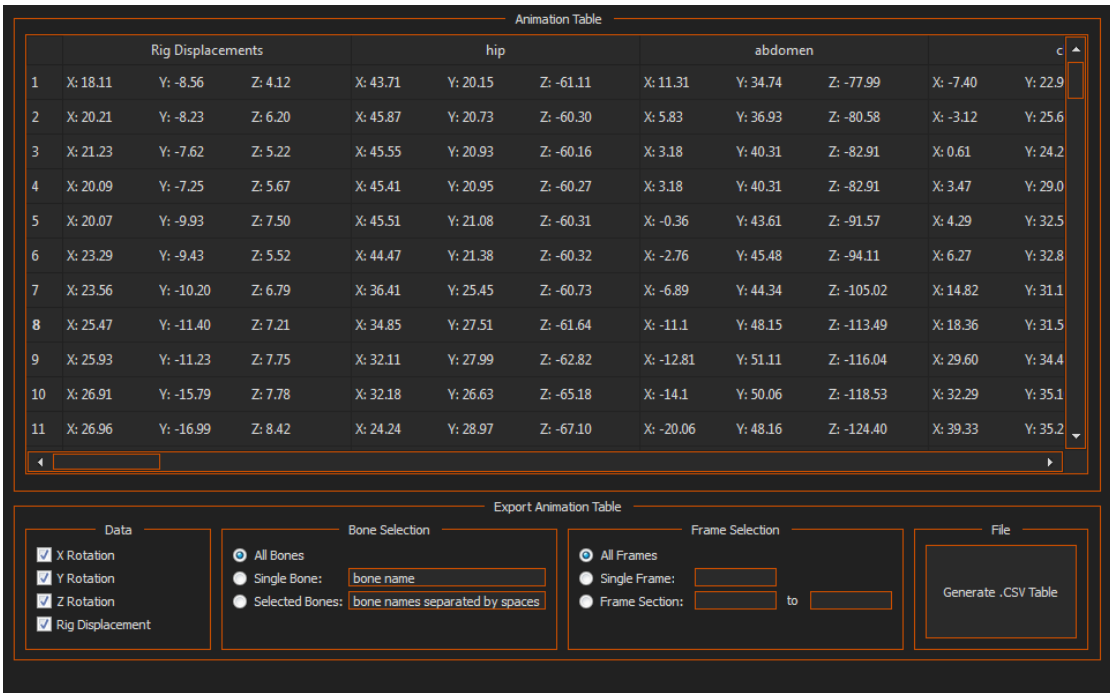
Magnetometer: X: 0 Y: 0 Z: 0

Fused: X: 0 Y: 0 Z: 0

Figure 3-9: SystemViewer illustrating the information of a hip IMU.

3.6.4 AnimationViewer

As shown in Figure 3-10, the last of the three viewers, the AnimationViewer, gives users access to the finalized motion. The resulting data is appropriate for behavioural experimentations or animating virtual characters. Unlike the previous two viewers, this interface is static and is generated after a recording session has terminated or when a pre-recorded animation is loaded. Motion capture files usually display readings as undecipherable blocks of numbers. This interface displays data in a coherent table where each column corresponds to a kinematic bone's rotation channel and each row corresponds to a frame of motion. The first column will always contain rig displacement values. The animation data can be exported by employing the widely supported yet simple comma-separated values (CSV) file format. Users have the option of exporting precise blocks of motion by specifying the axis of rotation, the kinematic channels and the frame intervals. This feature allows researchers to focus their attention only on the significant data while ignoring unneeded values.



Animation Table

	Rig Displacements			hip			abdomen			c	
1	X: 18.11	Y: -8.56	Z: 4.12	X: 43.71	Y: 20.15	Z: -61.11	X: 11.31	Y: 34.74	Z: -77.99	X: -7.40	Y: 22.9
2	X: 20.21	Y: -8.23	Z: 6.20	X: 45.87	Y: 20.73	Z: -60.30	X: 5.83	Y: 36.93	Z: -80.58	X: -3.12	Y: 25.6
3	X: 21.23	Y: -7.62	Z: 5.22	X: 45.55	Y: 20.93	Z: -60.16	X: 3.18	Y: 40.31	Z: -82.91	X: 0.61	Y: 24.2
4	X: 20.09	Y: -7.25	Z: 5.67	X: 45.41	Y: 20.95	Z: -60.27	X: 3.18	Y: 40.31	Z: -82.91	X: 3.47	Y: 29.0
5	X: 20.07	Y: -9.93	Z: 7.50	X: 45.51	Y: 21.08	Z: -60.31	X: -0.36	Y: 43.61	Z: -91.57	X: 4.29	Y: 32.5
6	X: 23.29	Y: -9.43	Z: 5.52	X: 44.47	Y: 21.38	Z: -60.32	X: -2.76	Y: 45.48	Z: -94.11	X: 6.27	Y: 32.8
7	X: 23.56	Y: -10.20	Z: 6.79	X: 36.41	Y: 25.45	Z: -60.73	X: -6.89	Y: 44.34	Z: -105.02	X: 14.82	Y: 31.1
8	X: 25.47	Y: -11.40	Z: 7.21	X: 34.85	Y: 27.51	Z: -61.64	X: -11.1	Y: 48.15	Z: -113.49	X: 18.36	Y: 31.5
9	X: 25.93	Y: -11.23	Z: 7.75	X: 32.11	Y: 27.99	Z: -62.82	X: -12.81	Y: 51.11	Z: -116.04	X: 29.60	Y: 34.4
10	X: 26.91	Y: -15.79	Z: 7.78	X: 32.18	Y: 26.63	Z: -65.18	X: -14.1	Y: 50.06	Z: -118.53	X: 32.29	Y: 35.1
11	X: 26.96	Y: -16.99	Z: 8.42	X: 24.24	Y: 28.97	Z: -67.10	X: -20.06	Y: 48.16	Z: -124.40	X: 39.33	Y: 35.2

Export Animation Table

Data
- ✓ X Rotation
- ✓ Y Rotation
- ✓ Z Rotation
- ✓ Rig Displacement

Bone Selection
- ● All Bones
- ○ Single Bone: bone name
- ○ Selected Bones: bone names separated by spaces

Frame Selection
- ● All Frames
- ○ Single Frame:
- ○ Frame Section: to

File

Figure 3-10: AnimationViewer showing animation data in a tabular format.

3.6.5 Rendering 3D Skeletal Topologies

Data is visualised in 3D using a viewport system. As shown in Figure 3-8, the viewport contains a drawing canvas that uses OpenGL to render the kinematic model. The skeleton geometry is computed locally, before any information is sent through the OpenGL pipeline.

The kinematic hierarchy is traversed and each skeleton bone object is rendered in sequence. Each bone is rendered as a result of four computations, all of which are achieved at the software level:

- *Geometry Instantiation:* The default bone topology is drawn facing upwards with a length and rotation of zero. Six vertices and 8 polygons form the bone's geometry.

- *Scaling:* Each bone is adjusted vertically so that its length matches the corresponding body part of the motion performer. The vertical axis coordinate of the upper most vertex is set to equal the bone's length.

- *Rotation:* The bone's quaternion rotation is used to rotate the six vertices to assume the desired orientation.

- *Translation:* The hierarchy of the kinematic model is traversed in order to compute the global origin of each bone. Based on those origins, the topologies are translated to interlink the bones and form a skeleton.

Performing the geometry instantiation, scaling, rotating and translating within the software is computationally expensive. An alternative approach is to compute the graphics within the OpenGL pipeline using matrices, a process which is further discussed in Chapter 5 in the context of mobile computing where the computational resources of the device are very limited.

The graphics are computed within software to allow the user to observe and potentially modify each stage of computation. As previously discussed the aim of this software environment is to provide a level of transparency to the researchers.

3.7 Concurrency and Software Timers

To achieve real-time motion capture streaming, concurrency and software timers become very important. Concurrency and software timers have two very important roles. First, due to the multicore architecture of modern CPUs, concurrency helps the software environment perform computations more efficiently by distributing the workload.

During recording sessions, the software environment fires two main concurrent threads, namely an interface thread and a back-end thread. Both threads employ timers to iterate through the code at different speeds. The interface thread focuses on GUI updates such as rendering or listening for user inputs. The back-end thread is aimed at computing motion as quickly as possible. Consequently, it is given priority over the interface thread to ensure that no data is lost or misinterpreted. This approach was chosen to prevent performance problems such as the interface slowing down the back-end processes.

The interface thread cannot compute numerous computationally expensive interface updates and render 3D visualisations at the same time. Consequently, another thread is assigned to render the viewport. Figure 3-11, illustrates the three main threads.

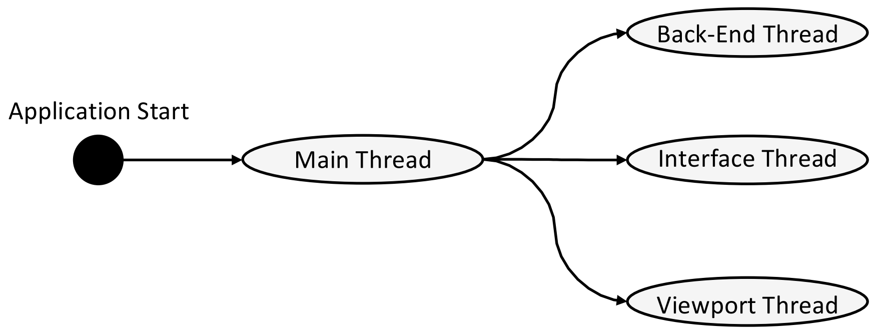


Figure 3-11: Skeletrix software environment multithreading diagram.

3.8 Motion Capture Reanimation

Motion capture reanimation is a term used by this thesis to describe a more efficient solution for inputting and outputting data within the Skeletrix software environment and to configure the software environment for use within a recording session. Existing software solutions require actor data, drivers, skeletal rigs and other files to be inputted by the user and configured manually before the system is ready to be used, which is a time consuming process. The fundamental concept behind motion capture reanimation is to combine the configuration files within one singular file format, which is referred throughout this thesis as BVHE. The advantage of BVHE over any existing format is practicality (simplifying the motion capture workflow) as users only have to load one file into the software environment. Given that kinematic hierarchy is driven by a system, which contains a system configuration, it is both logical and necessary to form a relationship between the two by establishing one singular and more complete file format.

In practice, the motion capture reanimation concept will becomes apparent shortly after being prompted by the user interface. Unlike other software environments that display pre-defined skeletal rigs upon start-up, the Skeletrix software environment stores no skeletal information natively. The user is asked to import a BVHE that automatically generates kinematic models, weight models, system models and software configurations. BVHE is the novel file type containing all these configuration attributes. This approach has a major impact on the software's usability and simplifies the number of steps taken by a user when commencing a recording session. Figure 3-12 depicts the contrast between the Skeletrix software environment and other software environments.

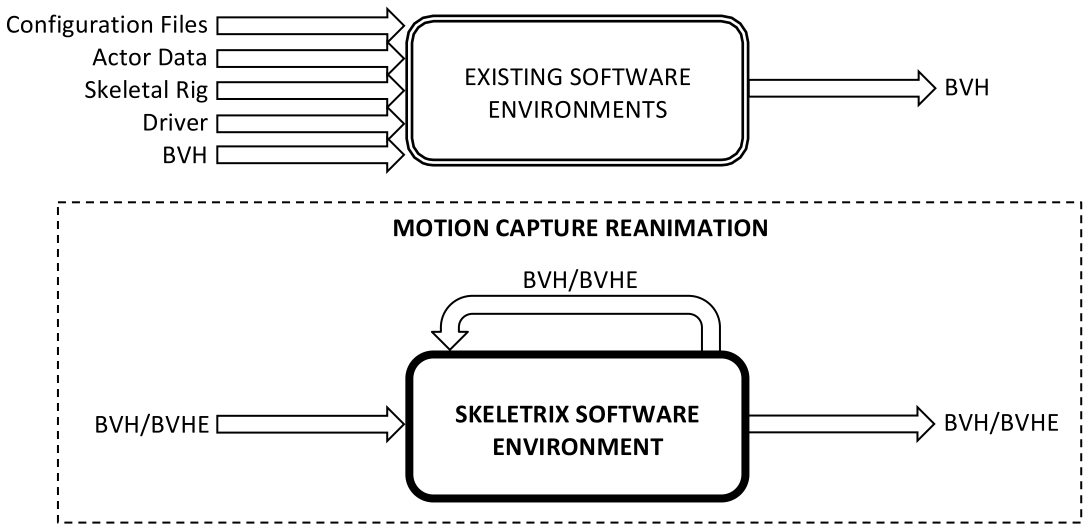

Configuration Files
Actor Data
Skeletal Rig
Driver
BVH
EXISTING SOFTWARE ENVIRONMENTS
BVH
MOTION CAPTURE REANIMATION
BVH/BVHE
BVH/BVHE
SKELETRIX SOFTWARE ENVIRONMENT
BVH/BVHE

Figure 3-12: An overview of motion capture software environments highlighting the concept of motion capture reanimation and usability simplification through BVHE.

BVHE replaces the need for multiple configuration files with one singular input. Once a recording session is completed, the resulting BVHE file will store all the configuration information of that session. Therefore, it can be used at a later date as an input to configure the system for reuse.

3.9 Biovision Hierarchy Extended

The standardization of file formats is a frequent requirement in software development as it promotes interoperability and information sharing. The lack of a standard implies an industry that has too many incomplete solutions competing for software support. For this reason, introducing an entirely new format, in an otherwise saturated pool of solutions, can be counterintuitive as developers are not eager to learn or implement new file systems. A possible approach is to take an existing format, that developers are already familiar with, and extend it to suit the needs of the industry.

This section introduces Biovision Hierarchy Extended (BVHE), a novel file type designed to simplify the user/system interaction during recording sessions. BVHE is an extended version of the Biovision Hierarchy (BVH) format. The new syntax allows the file type to store system specific data such as computer-hardware connection parameters and the sensor configuration parameters. It removes the need for actor file systems by repurposing BVH's skeletal definitions to reflect the user's bodily proportions. BVHE is an integral requirement and component for the Skeletrix software environment architecture.

3.9.1 Biovision Hierarchy Format

Given that BVHE encapsulates standard BVH code, it is appropriate to review the original format. BVH first establishes a skeletal rig consisting of multiple bone definitions and one kinematic hierarchy. A bone definition is characterized by three properties: length, orientation and channels. The length and orientation can be expressed as one positional offset where the vector distance is the length and the vector direction is the orientation. Notably, this approach is not good at defining a skeletal segment's roll. The roll is often irrelevant if the rig is rendered as three-dimensional lines or cylinders. Unlike other motion capture formats, BVH allows bone definitions to contain multiple offsets whereby one skeletal segment can have one origin and multiple endings. The channels specify three empty variables through which a transformation sequence of angles can be assigned the skeletal segment in question. Next, the hierarchy element comes into play by constructing the skeleton. Indenting bone definition within brackets, whereby the indent value and bracket count symbolize the hierarchical depth, is an effective method for representing the hierarchy links. The visual reconstruction phase uses that information to reposition each bone so that its origin assumes the position of its parent's end point. Finally, the motion data is placed at the end of the file as a list of floating-point values in accordance to the channel template. Most motion capture file formats specify the number of animation frames to simplify parsing.

3.9.2 Biovision Hierarchy Extensions

The new BVHE extensions are inserted in the middle of a standard BVH file, between the kinematic hierarchy specification and the motion data. To begin with, the new semantics help the software environment identify the appropriate driver for interfacing with a particular device. Then, both software and hardware are readied for intercommunication by defining the sensors, data channels and data properties. Delineating all system properties within an editable file allows users to customize the way in which they use motion capture devices. The BVHE extensions are denoted by the following attributes.

- *DLL:* Once a software environment parses the motion file, the dynamic-link library (DLL) attribute specifies which driver to be loaded by default. This property should be editable through the interface. The user may be experimenting with various motion capture devices simultaneously. Stating the driver keeps track of what system should be used with this particular configuration.

- *Port:* Operating systems use ports to communicate with peripherals through serial, USB or Bluetooth. Once the connection is acknowledged, each device is assigned a port value so that it may be accessed by software applications. Ports can vary depending on the machine in use. This attribute specifies a default port.

- *Versioning:* If the user is developing drivers, this attribute is important in keeping track of development releases.

- *Type:* Although the Skeletrix software environment is primarily designed to support inertial systems; other technologies could also be integrated. This attribute informs the software environment if the system is inertial, exoskeleton or optical. Skeletrix anticipates rotational data from inertial/exoskeleton hardware and positions from optical devices. The default type is inertial.

- *Space:* All rotations and positions can be expressed in world or local space. Devices containing multiple sensors may calculate inverse kinematics at a firmware or driver level. The software environment needs to know whether to calculate the kinematics or directly apply motion data to the skeleton. The default space is world.

- *Sensor:* A BVHE file may contain multiple sensor definitions. The user may be experimenting with a single IMU or a more complex BSN. Sensor definitions are similar to kinematic hierarchies but linear. Each sensor controls the transformations of one or, through the kinematic constraints of the rig, several bone entities. Each sensor contains an optional rotational offset and a scale value. Although these two attributes are not implemented in the software environment, they are desirable for user profiling to calibrate the rig more accurately to match the motion performer’s body proportions.

3.9.3 Biovision Hierarchy Extended Syntax

The following syntax, shown in Figure 3-13, constitutes a basic BVHE file by first defining a BVH skeletal hierarchy encompassing three bones: chest, neck and head. Three channels are assigned for person dead reckoning while the following nine define the roll/yaw/pitch (in that specific order) of each bone. Therefore, each animation frame will require twelve numeric entries consisting of positional and angular readings. The BVHE extensions prepare the software environment for intercommunications with an inertial system encompassing a two-sensor configuration. That device utilizes the standard dynamic-link library to receive information from communication port three. The IMU naming scheme suggests hardware containing IMU3000 gyroscopes and MMA8450 accelerometers, matching the hardware presented in Chapter 4. This particular inertial device computes no inverse kinematics and outputs world space rotations. The software environment will receive three numeric values from each sensor corresponding to yaw, pitch and roll.

```
HIERARCHY

ROOT Chest {
  OFFSET 0 0 0
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT Neck {
    OFFSET 0 20.6881 -0.73152
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT Head {
      OFFSET 0 11.7043 -0.48768
      CHANNELS 3 Zrotation Xrotation Yrotation
      END SITE {
        OFFSET 1 0 0
      }
    }
  }
}

SYSTEM

DLL default.DLL
PORT 3
VERSIONING v1.0
TYPE inertial
SPACE world

SENSOR imu3000_mma8450_1 {
  BONE Chest
  OFFSET 11 10 0
  SCALE 110
}

SENSOR imu3000_mma8450_2 {
  BONE Head
  OFFSET 0 0 0
  SCALE 100
}

MOTION

FRAMES 3
FRAME TIME 0.00833333
52.122 85.152 1.6616 0.4511 -0.0026 0.0129 0.0 0.0 0.0 0.1245 -0.004 0.0215
51.529 84.612 2.2768 0.9193 -4.1156 -3.465 0.0 0.0 0.0 0.1900 0.1221 -0.171
51.528 84.551 2.1411 0.9276 -4.1653 -3.288 0.0 0.0 0.0 0.2635 0.2667 -0.376
```

Figure 3-13: BVHE syntax denoting a three-bone hierarchy connected to two sensors.

3.9.4 File Parsing

Because BVH and the new file format BVHE (published in Pascu et al. [17]) are somewhat similar, a shared parser was developed to support both file types. As previously mentioned, supporting BVH was important for cross-compatibility reasons. The Skeletrix software environment parser treats all files in the same fashion until BVHE syntax is identified. Figure 3-14 is an activity diagram that illustrates the entire parsing process whereby BVH/BVHE data is inputted to produce a kinematic model and an animation model that stores motion data.

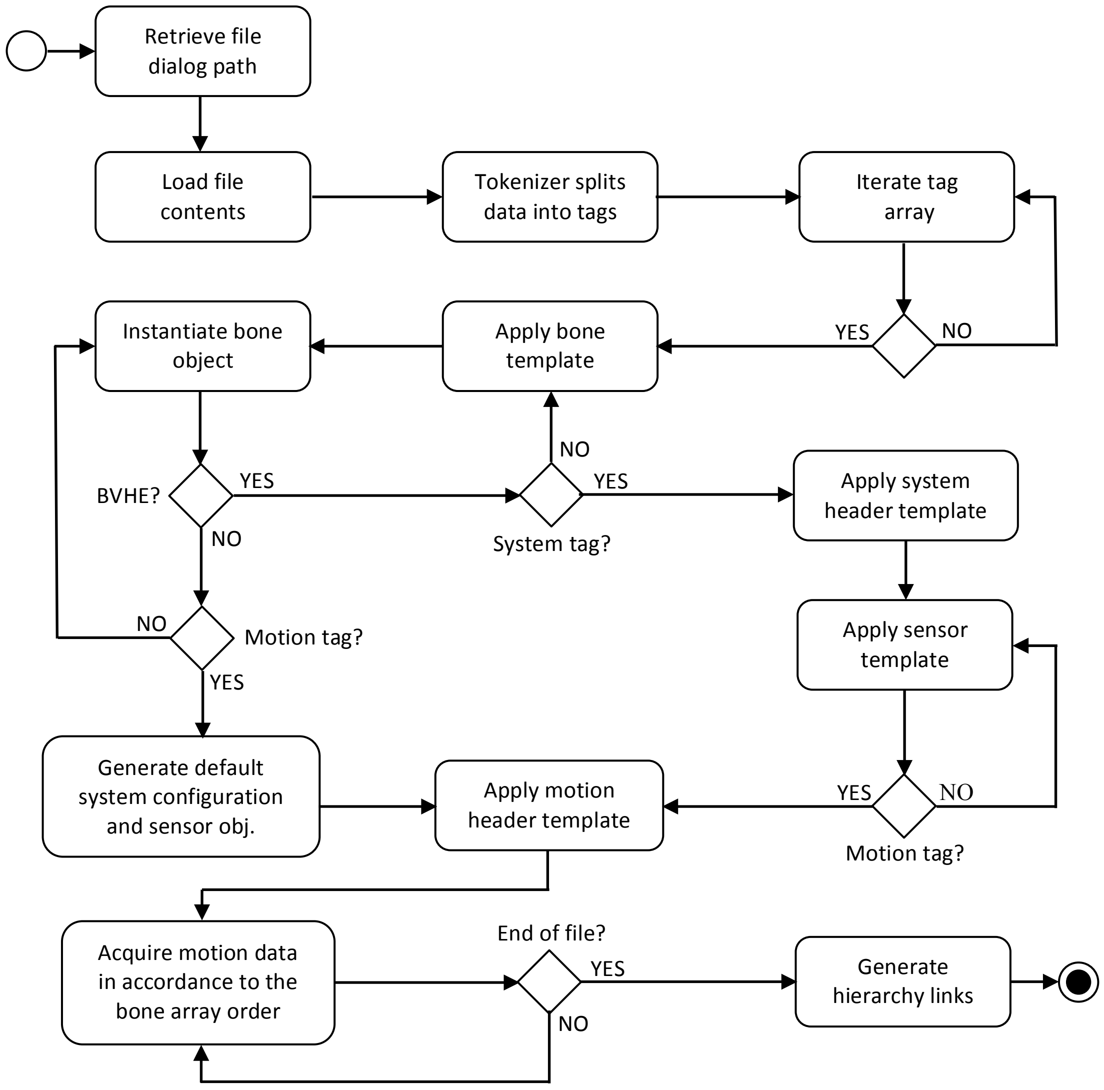

Retrieve file dialog path
Load file contents
Tokenizer splits data into tags
Iterate tag array
YES
NO
Apply bone template
Instantiate bone object
BVHE?
YES
NO
System tag?
Apply system header template
Motion tag?
Apply sensor template
Generate default system configuration and sensor obj.
Apply motion header template
Motion tag?
Acquire motion data in accordance to the bone array order
End of file?
Generate hierarchy links

Figure 3-14: BVH/BVHE file parsing activity diagram.

3.10 Driver Development Kit

Most motion editing packages provide methodologies for developing small plugins that suit the specific needs of users. Generally, a plugin is a small software application that embeds a new tool or feature within the software environment's interface. Unlike motion editing packages, this software environment's primary focus is to acquire and process motion data from hardware. Instead of plugins, it provides the capability to develop and import device drives. In this context, a driver is a small software application, enclosed within dynamic-link library (DLL), used to interface with external hardware and retrieve motion readings safely. Example source code of a driver can be seen in Appendix B.

The driver development kit (DDK) provides researches with a bridging architecture for connecting to inertial motion devices. The underlining purpose is to motivate potential users to develop sensor devices by providing the tools to connect them to a computer. The multifaceted sequences of operations for gathering, refining and outputting usable motion capture data are provided as part of Skeletrix. Within the scope of this thesis, this DDK allows for experimentations that employ a wide range of motion capture devices such as the Shimmer R2, Razor Attitude Heading Reference System (AHRS) and the Motion Tracking Development System (MTDS) suit presented in the next chapter, Chapter 4.

Enclosing drivers in DLLs can simplify the number of user-performed actions in connecting to a system. A major benefit is that DLLs do not require operating system installations. Notably, a significant limitation is that DLLs can only operate on Windows-based machines. This approach provides a level of modularity where one computer could be used to interface several dissimilar devices (or one device in different ways) by slotting in and out driver modules.

Before enabling data intercommunications, the first step of the bridging procedure involves driver validation through which a connection is established between the software environment and driver. The system manager interrogates the driver and validates its attributes to ensure a trouble-free recording session. The sequence diagram depicted in Figure 3-15 illustrates the list of questions (or software interrogations) addressed to the driver.

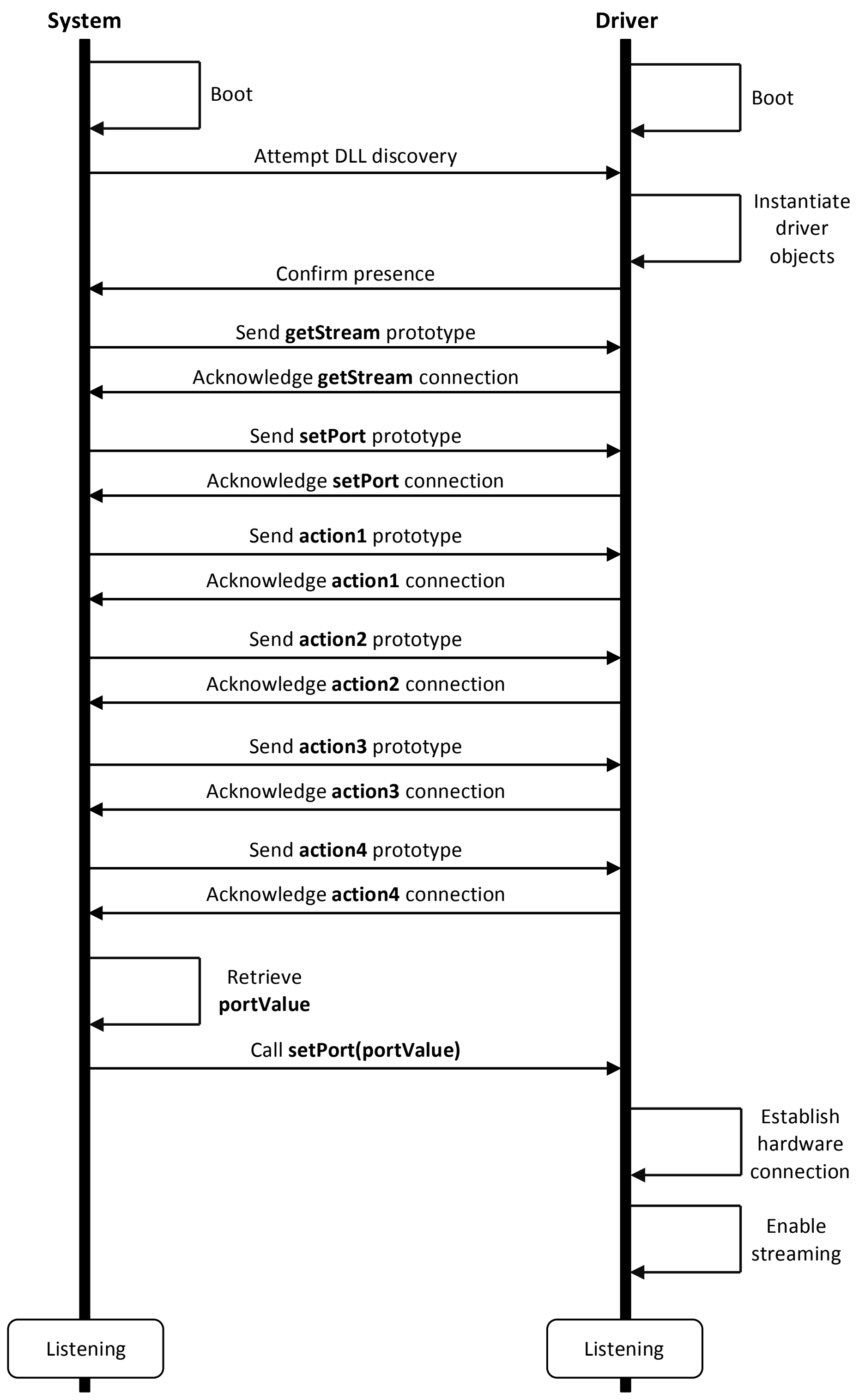


Figure 3-15: Driver handshake sequence diagram.

Function prototypes are used to access and trigger driver code. The driver template contains a specific list of function prototypes: *getStream*, *setPort*, *action1*, *action2*, *action3* and *action4*. All must be present, even if redundant, for the driver connection to be validated. The four actions may be used by the more complex full body suits that require intermediate steps for hardware initialization. For example, the Animazoo IGS suits require additional handshaking and Northing for the magnetometers to be reset and tuned to face north. Northing could be assigned to one of the four actions and the corresponding code could be included in the driver.

Having established a valid connection, the next step is data streaming whereby data is extracted from hardware and visualised in real-time. Each driver must have a buffer that fills up with angular readings. Motion capture recording sessions may require fast frequencies that put a lot of pressure on the connection between the computer and hardware. While a buffer will induce latency, a buffer is needed to relax that connection and ensure that no data is lost. The following sequence diagram, shown in Figure 3-16, illustrates the data acquisition process. The transmission rate, at which data buffers are received from the driver, is generally 30f/s. This speed is pre-configured and required by the 3D engine to render smooth visualisations in real-time. For performance reasons, the 3D engine will only render the visualisation at this speed even if the transmission rate is faster.

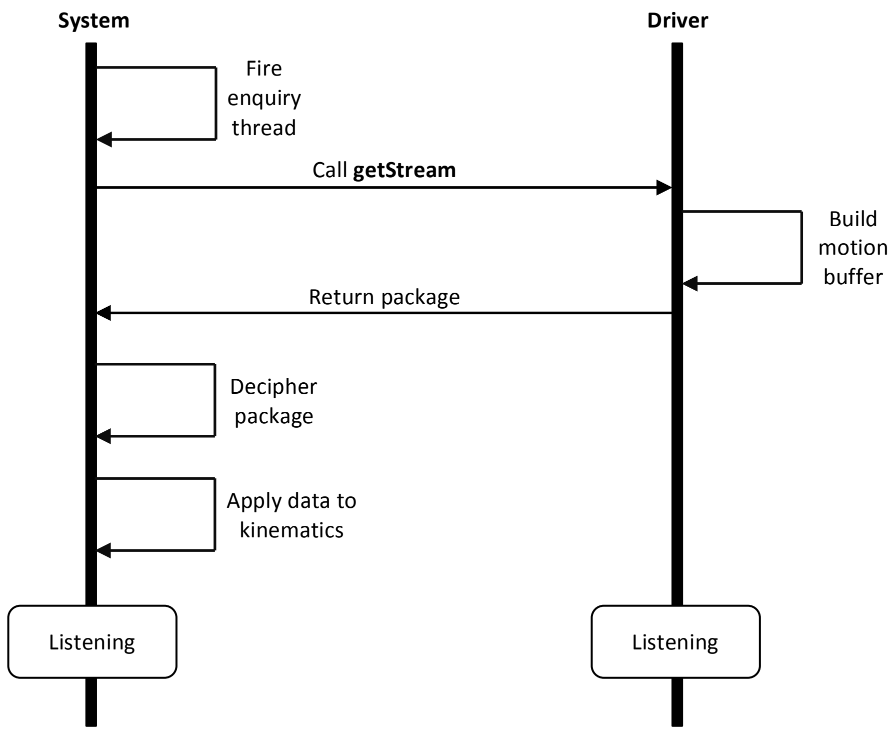


Figure 3-16: Driver data acquisition sequence diagram.

3.11 Versioning

The development of the software environment was achieved in incremental stages, in parallel with the development of the MTDS hardware presented in Chapter 4. As shown in Table 3-1, this section discusses the five main release versions.

The first version was centred on creating a kinematic model and visualising raw motion data in 3D. It featured the core libraries for vector and quaternion mathematics, the interface layer (that can render OpenGL graphics) and a file parser for BVH data. The next three versions expanded on: the concept of having viewers (AnimationViewer, KinematicsViewer and SystemViewer), the DDK, BVHE support and hardware integration. The development of the final version added Internet connectivity to interact with the Motion Cloud.

Table 3-1: Skeletrix software environment versioning.

Functionality	**1.0 – 1.5**	**2.0 – 2.8**	**3.0 – 3.10**	**4.0 – 4.10**	**5.0 – 5.17**
Core Functionality					
Vector Library	**YES**	**YES**	**YES**	**YES**	**YES**
Quaternion Library	**YES**	**YES**	**YES**	**YES**	**YES**
OpenGL Renderer	**YES**	**YES**	**YES**	**YES**	**YES**
QT Interface Layer	**YES**	**YES**	**YES**	**YES**	**YES**
Multimodal Viewports	N/A	**YES**	**YES**	**YES**	**YES**
Data I/O (BVH)	**YES**	**YES**	**YES**	**YES**	**YES**
Data I/O (BVHE)	N/A	N/A	**YES**	**YES**	**YES**
Local Storage of Motion	N/A	**YES**	**YES**	**YES**	**YES**
Models					
Kinematic Model	**YES**	**YES**	**YES**	**YES**	**YES**
Animation Model	N/A	**YES**	**YES**	**YES**	**YES**
System Model	N/A	N/A	**YES**	**YES**	**YES**
Weight Model	N/A	N/A	**YES**	**YES**	**YES**
Viewers					
KinematicsViewer	N/A	**YES**	**YES**	**YES**	**YES**
AnimationViewer	N/A	N/A	**YES**	**YES**	**YES**
SystemViewer	N/A	N/A	N/A	**YES**	**YES**
MotionCloudViewer	N/A	N/A	N/A	N/A	**YES**
Driver Development					
MTDS	**YES**	**YES**	**YES**	**YES**	**YES**
Shimmer R2	N/A	N/A	**YES**	**YES**	**YES**
Razor AHRS	N/A	N/A	N/A	**YES**	**YES**
Other					
DDK	N/A	N/A	**YES**	**YES**	**YES**
Web Connectivity	N/A	N/A	N/A	N/A	**YES**

3.12 Lowest-Anchor Centre of Mass Algorithm

Dead reckoning is the process of computing horizontal displacement over longer periods of time for a kinematic model. When taking a step, the body is propelled forward by a distance equal to the stride length. After a sequence of steps, several stride lengths are summed to determine the person's position in space. Inertial motion capture systems have no inherent sensors to measure that displacement. While additional ultrasound or optical sensors can be used to aid the inertial system in computing dead reckoning, this solution overcomplicates the process. The simplest and possibly most problematic methodology, in terms of accuracy, for achieving this task is the lowest-point algorithm [127]. The lowest-point algorithm is an effective yet simple method of applying planar collision detection and inverse kinematics algorithms to a skeletal rig to compute dead reckoning.

As previously published in Pascu et al. [18], the lowest-anchor centre of mass algorithm (LACOMA) is a more complex solution, using the body's musculoskeletal centre of mass, aimed at computing the anchor of a kinematic model more accurately. The anchor of a skeletal rig is the support point on which the majority of the body's weight is balanced. Knowing which kinematic segment is impacting the floor is important. For example, if a person is dragging their feet, the lowest-point algorithm will prevent the skeleton from moving. This problem can be found in a variety of gestures. In essence, the LACOMA study is aimed at finding whether the musculoskeletal axis of balance is beneficial in computing better anchors during gait.

3.12.1 Understanding the Lowest-Point Algorithm

But what is the lowest-point algorithm? As the title would imply, the lowest point (or kinematic segment) of the kinematic model is likely to be the supporting point of the body. This solution would be ideal in a scenario where the motion is recorded on a flat surface and the motion capture system is perfectly accurate. While gyroscopes drift, accelerometers and magnetometers produce noise. These inaccuracies cause the kinematic model to misrepresent the body. Additionally, it is likely that the kinematic model is not matching the motion performer in terms of proportions. Representing the otherwise complicated anatomy of the human body, as a simplistic list of approximately 40 bones, will produce even more inaccuracies. The body's asymmetries are not often replicated in the kinematic model. Solutions for solving this problem at the software level are often protected, or even patented, by the system manufacturers. The most common solution is the lowest-point algorithm because it is computationally inexpensive and straightforward to implement.

During one ambulatory step, the anchor swaps between feet as the body switches from one foot to the other. Figure 3-17 illustrates how that swap occurs in three steps (this process has previously been discussed in Section 2.7.4 of the literature survey). The anchor swaps from the blue leg to the green leg.

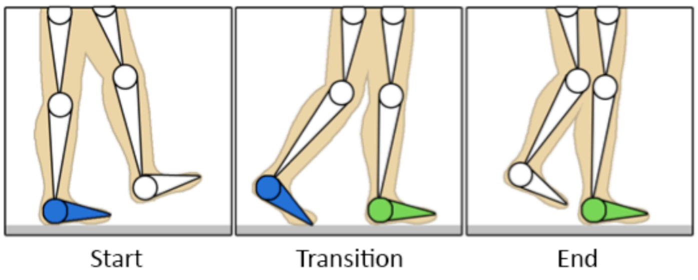


Figure 3-17: Anchor swapping during one ambulatory step. **Source: Pascu et al. [18]**

Motion capture in a recording studio usually involves characters performing gestures on a flat surface. Although the lowest-point algorithm usually serves the purpose of computing kinematic displacement, it proves problematic in scenarios where the motion performers are not taking clear steps. For example, if a person is dragging their feet or shifting balance between feet while stationary, the algorithm will not work because the system is likely to miscalculate the correct foot. These errors are often corrected during data cleaning, a time consuming process where an animator corrects the recorded motion manually. Figure 3-18 shows how the software-level confusion occurs because the anchor selection becomes ambiguous.

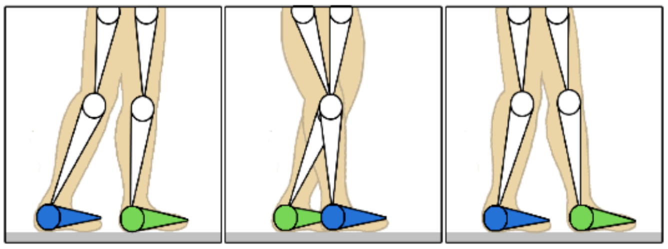


Figure 3-18: Confusion during the anchor selection when the motion performer is dragging their feet. **Source: Pascu et al. [18]**

3.12.2 Determining a Weight Model

The first stage of LACOMA is methodology for adding physical properties to the kinematic model through a weight model. The weight model will later be used to compute the body's musculoskeletal axis of balance. A weight model consists of positional offsets and weight values that are supplemented to a kinematic model, thus forming a primitive physics engine. Figure 3-19 shows an example of a weight model in the software environment, as used throughout the experiment. On the left (1), the kinematic model is rendered with bone geometry. In the middle (2), the kinematic joints are rendered alongside centres of weight. On the right (3), the weight model is render by itself.

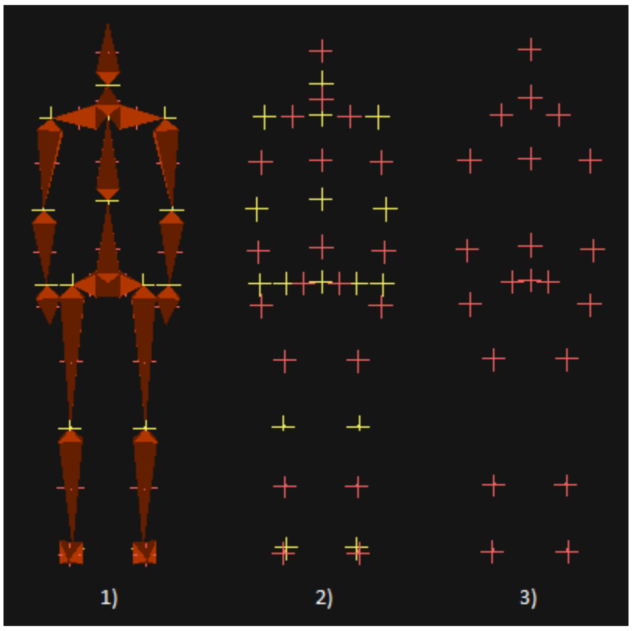


Figure 3-19: Weight model rendered using the software environment. **Source: [18]**

A weight model is constructed by adding a balance ratio and a weight value to each bone in the kinematic skeleton. The balance ratio determines a skeletal segment's the centre of weight using the tip and base. For example, a skeletal segment having a length of 10 units and balance ratio of 45% will have a centre of balance offset of 4.5 units resulting in a base-heavy bone such as the bicep (which is heavier at the shoulder joint than it is at the elbow joint). A weight ratio greater than 50% will describe a tip-heavy bone. Each skeletal segment must be given a scalar coefficient that is less than 1 so that the combined weight of the model equals to 1.

The body generally becomes thinner and lighter towards its extremities. For example, the torso is the heaviest section, followed by the arms and shins, forearms and calves, hands and feet. In accordance to the kinematic model shown above, most of the skeletal segments will be base-heavy. As shown in table 3-2, the weight distribution throughout the body was determined by looking at studies focused on human anatomy such as [143]. Complex joints (e.g. shoulders, hips) were given an even distribution.

Table 3-2: Weight model balance ratios and weight values. **Source: [18]**

Segment	Weight	Tip Distance	Base Distance
Upper Body			
Abdomen	0.09007	42.4%	57.6%
Chest	0.11778	46%	54%
Neck	0.03023	51%	49%
Head	0.06928	50%	50%
Left Shoulder	0.02887	50%	50%
Right Shoulder	0.02887	50%	50%
Left Arm	0.04850	50.7%	49.3%
Right Arm	0.04850	50.7%	49.3%
Left Forearm	0.03811	56.6%	43.4%
Right Forearm	0.03811	56.6%	43.4%
Left Hand	0.03579	54.2%	45.8%
Right Hand	0.03579	54.2%	45.8%
Lower Body			
Left Hip	0.02887	50%	50%
Right Hip	0.02887	50%	50%
Left Thigh	0.07737	53.5%	46.5%
Right Thigh	0.07737	53.5%	46.5%
Left Shin	0.06120	67.4%	32.6%
Right Shin	0.06120	67.4%	32.6%
Left Foot	0.02811	47.6%	52.4%
Right Foot	0.02811	47.6%	52.4%

3.12.3 Computing the Anchor

This section discusses the computation of the skeleton's anchor point using the weight model and an axis of balance. Figure 3-20 illustrates the pseudo code for the algorithm, which works in three different stages. For the purpose of this algorithm, let's establish that the vertical axis is Y.

The first stage is to compute anchor candidates. Unlike the lowest-point algorithm that considers only one anchor, LACOMA first defines a set of multiple anchor candidates. A threshold plane is placed within the proximity of the ground, approximately at the height of the tibia. Any bone intersecting that plane will be considered an anchor candidate. The intersection is calculated by simply verifying the vertical position of the bone. If the base, the tip or both are below the threshold, the bone is deemed an anchor candidate as its anchor flag is set to true.

The second stage is to compute an axis of balance, which is a vertical vector defined by an *x* and *z* coefficient. The pseudo code traverses the skeleton and multiplies every positional centre of weight (only the *x* and *z* coefficients) by the scalar weight value. The result is concatenated to an array and the mean average of the array will produce the axis of balance.

The third stage of the algorithm is to compute the correct anchor (or supporting bone) of the body. The algorithm iterates through the skeleton, selects the anchors and calculates the distance between each bone and the axis of balance. The weight model stores distances as proximity values. The skeletal segment with the lowest proximity value will be chosen as the final anchor.

```
INITIALIZE temporary array ax AS empty
INITIALIZE temporary array az AS empty
INITIALIZE axis of balance AS vector {0,0}
INITIALIZE anchor AS blank bone pointer

// STAGE 1: DETECT ANCHORS CANDIDATES
FOR EACH bone IN skeleton
       IF bone tip < threshold AND/OR bone base < threshold
              SET bone anchor TO true
       END IF
END FOR EACH

// STAGE 2: COMPUTE AXIS OF BALANCE
FOR EACH bone IN skeleton
       CONCATENATE (bone centre of weight x * bone weight)
              TO temporary array ax
       CONCATENATE (bone centre of weight z * bone weight)
              TO temporary array ay
END FOR EACH
SET axis of balance x TO MEAN AVERAGE OF temporary array ax
SET axis of balance z TO MEAN AVERAGE OF temporary array az

// STAGE 3: FIND LOWEST ANCHOR
FOR EACH bone IN skeleton WHERE anchor = true
       SET bone proximity
        TO DISTANCE FROM axis of balance TO bone centre of weight
       IF bone proximity < previous bone proximity
              SET anchor TO bone
       END IF
END FOR EACH
RETURN anchor
```

Figure 3-20: Computing the lowest anchor of a skeleton using a weight model.

3.12.4 Simulations

As published in Pascu et al. [18], the algorithm was tested through five simulations with the aim of demonstrating that the anchor is computed correctly. Five pre-recorded gestures were loaded where the individual is: dragging their feet, shifting balance between legs, kneeling, crawling and going into prone position. Snapshots were taken of sections where the lowest-point algorithm would otherwise miscalculate the correct anchor.

Table 3-3 illustrates the results of the simulations where LACOMA is used to compute a set of anchor candidates, an axis of balance and the proximity of each bone to the axis. The algorithm computes the proximity values to determine the correct anchor whereby the candidate with the lowest proximity value wins. These results demonstrate the proposed algorithm functioning correctly.

Table 3-3: LACOMA simulations computing the correct anchor. **Source: Pascu et al. [18]**

Desired Anchor	Anchor Candidate	Distance to Axis of Balance
Gesture 1: Dragging Feet		
Left Foot	**Left Foot**	**3.447140376**
	Right Foot	7.360259044
Gesture 2: Balance Shifting		
Left Foot	**Left Foot**	**2.634964232**
	Right Foot	4.951058102
Gesture 3: Kneeling		
Left Foot	**Left Foot**	**2.911940236**
	Right Foot	4.033130481
	Left Shin	4.607794689
	Right Shin	5.749732326
Gesture 4: Crawling		
Left Shin	Left Foot	6.1504404348
	Right Foot	6.3562048703
	Left Shin	**2.4627770625**
	Right Shin	3.5748058548
	Left Hand	15.7069667215
	Right Hand	13.5998812961
Gesture 5: Prone Position		
Right Shin	Left Foot	19.0994519007
	Right Foot	18.0061532627
	Left Shin	12.8933432307
	Right Shin	**11.5659549656**
	Left Hand	23.8645977278
	Right Hand	22.3093305126
	Left Forearm	20.2661181134
	Right Forearm	18.5971782273

3.13 Conclusions

This chapter has delineated the design and implementation of a software environment with two distinct goals. The large-scale goal is to provide a suitable simulation environment for motion capture experimentations to encourage users and researchers to join the field of inertial motion capture and BSNs. The Skeletrix software environment is an object-oriented tool for data gathering, processing and scientific analysis that employs multiple software development domains: concurrent programming, interactive visualizations, language processing, rendering, etc. It envelops methodologies for interfacing with a wide range of inertial and exoskeleton systems with potential (if developed further) for optical systems. Users can produce drivers to integrate various motion capture devices by using the DDK in conjunction with BVHE.

In an effort to expand its spectrum of application areas, this software environment does not store any kinematic models, system models or motion data natively. By placing all those elements in an external configuration file (i.e. BVHE), Skeletrix provides a high level of customizability. All skeletal rig constructions and hardware configurations tasks are achieved by simply editing BVHE code. The requirements specification introduces the principle of transparency whereby users are given full access to observe and visualize the software's back-end through its interface. Data can be accessed at three levels using the KinematicsViewer, SystemViewer and AnimationViewer accordance to the open source paradigm, this software environment also provides the programming scaffolding to support user-produced additions. The fourth viewer, the MotionCloudViewer, is further discussed in Chapter 6 in the context of the Motion Cloud.

As identified in Chapter 2, file format standardization is a problem that limits software cross-compatibility and information sharing. The field of motion capture presents a saturated pool solutions competing for software support. This chapter has introduced a new approach to standardizing file systems. Instead of creating an entirely new language, which would require extensive implementations throughout the industry, BVHE repurposes existing BVH code. Applying existing file systems to solve reoccurring problems is an efficient approach that promotes existing software cross-compatibilities. The BVHE format includes hardware configuration data to streamline the user’s interaction with hardware. As previously discussed, the main advantage of BVHE over any existing format is practicality by simplifying the motion capture workflow. Users only have to load one file into the software environment to start using an inertial motion capture system. Given that kinematic hierarchy is driven by a system, which contains a system configuration, it is both logical and necessary to form a relationship between the two by establishing one singular and more complete file format. The ultimate goal is to introduce plug-and-play simplicity of computer peripherals in the context of more complex inertial suits.

This chapter has showcased the design and development of the DDK, a flexible approach for integrating inertial hardware with software. The DDK is designed to be modular whereby a module is a self-contained DLL. Each DLL must be built in accordance to a predefined template, which defines the bridging procedure between the two applications. The DDK will be further evaluated throughout the next chapters through the development of a motion capture suit and the integration of several sensor devices.

This chapter has demonstrated how the Skeletrix software environment can be used to test and evaluate the lowest-anchor centre of mass algorithm (LACOMA), a solution for computing the supporting anchor of a kinematic model during gait. The proposed solution introduces the concept of a weight model to add basic physics to a kinematic model. The weight model is used to compute the axis of balance of the body. Using the axis of balance it the kinematic anchor of the skeleton can be estimated more accurately. While the algorithm shows potential through a set of five simulations, there is room for further development to integrate this solution throughout the Skeletrix framework.

3.13.1 Application Areas

This section introduces a set of potential application areas for the developments presented in this chapter. The proposed software environment represents a tool for developing inertial motion capture systems and is used throughout the remainder of this thesis. But first, let’s consider three additional scenarios: hardware development, constructing heterogeneous BSNs and benchmarking.

Scenario 1: Hardware Development

The development of an inertial motion capture system requires computer-side software, as rotational data cannot be used in the context of character animation without kinematics. Throughout this chapter, emphasis was put on the DDK and BVHE to demonstrate a better approach for creating computer-hardware intercommunications. The Skeletrix software environment has the potential to be used as a tool for developing hardware as it simplifies the procedure of extracting and processing rotational data. The SystemViewer can be used to analyse the raw system data, the KinematicsViewer can be used to visualise the character motion and the AnimationViewer can be used to output the data for further analysis. This usability scenario is demonstrated throughout the next chapter, Chapter 4, in the context of MTDS.

Scenario 2: Constructing Heterogeneous BSNs

Inertial motion capture systems imply the use of an array of sensors. It is difficult to construct a BSN without acquiring an otherwise expensive motion capture system. The concept of a heterogeneous BSN, previously published in Pascu et al. [17] and previously discussed in Section 2.6.2, has the potential to make inertial motion capture system more attainable to the average user. While the DDK allows for the easy integration of hardware, the BVHE file format is specifies the hardware configuration of the system in use. Using these unique features, several dissimilar sensors (that may differ in terms of hardware) could be interconnected to form a prototype level motion capture system. Heterogeneous BSNs represent a step forward towards making inertial motion capture systems more flexible.

Scenario 3: Benchmarking

A key aspect of the field of motion capture is benchmarking whereby several systems are compared to measure performance. The concept of performance is introduced throughout this thesis as a measure of the difference between recorded and real life motion. There are many sensor attributes (see Section 2.5.2) that are significant factors in achieving performance. A key aspect of benchmarking is to avoid being bias towards a system. At present, inertial motion capture hardware can be benchmarked using their bespoke software applications, which achieve their functionality using varying methodologies. For example, benchmarking the hardware inside two identical sensor devices, that use two software-side methods for calibration, will produce a bias. If one of the calibration methods is more successful than the other, the data will be compromised throughout the recording, thus making two identical systems have different performance attributes. Using the software environment presented in this chapter for both systems will remove that bias and make the benchmarking process more objective. This is because both system will use identical software.

CHAPTER FOUR

4 Constructing Inertial Body Sensor Networks

4.1 Introduction

The Skeletrix software environment, previously introduced in Chapter 3, provides a software layer for inertial motion capture systems while providing unique solutions to the challenges faced when acquiring and processing motion data produced by body sensor networks (BSN). The motion data obtained must be processed in a specific manner to become useful in the context of skeletal motion reconstruction.

While the previous chapter has focused on the software aspect, this chapter is centred on the integration of commercial hardware and the development of an entirely new BSN. As shown in Figure 4-1, the development of the new BSN comprises of both firmware and hardware development.

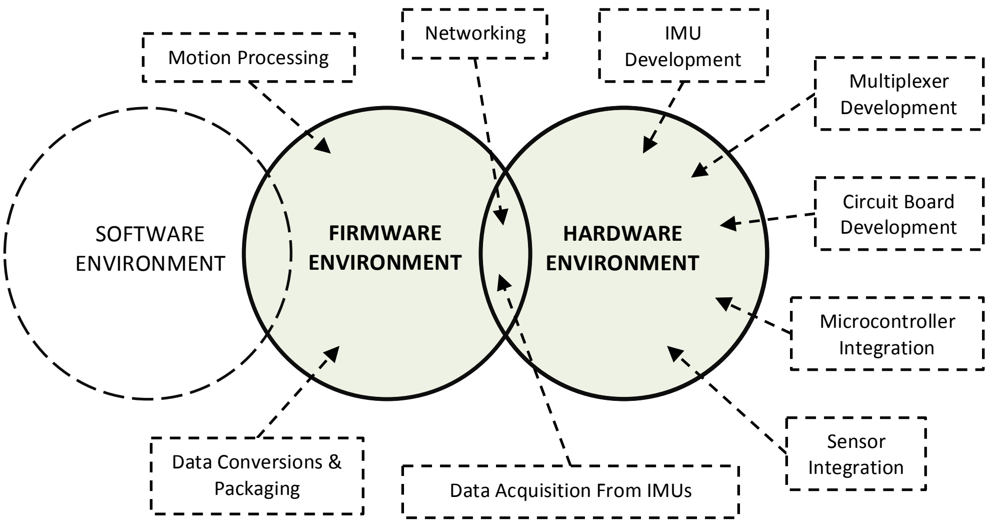


Figure 4-1: A high-level representation of motion capture frameworks with emphasis on the firmware and hardware aspects and their technologies.

In broad terms, hardware development involves printing circuit boards, integrating microcontrollers and integrating inertial motion sensors (e.g. gyroscopes, accelerometers and magnetometers). A BSN will consist of several inertial measurement units (IMU) connected to a multiplexer. Firmware development is concerned with programming the microcontrollers to extract motion data from the sensors, process it and communicate it across the network.

This chapter's first aim is to demonstrate the integration of hardware within the framework using the Skeletrix software environment. Two commercially available IMUs are integrated to form two case studies. The two IMUs provide a source of motion data that can be used throughout the framework (e.g. with the Motion Cloud which is published in Pascu et al. [15] [16] [19]). This chapter's second aim is to form a critical discussion on the topic of BSN development. The discussion is centred on the development of a prototype BSN entitled Motion Tracking Development System (MTDS) Pascu et al. [18].

MTDS is discussed with emphasis on the requirements specification, conceptualization and development. MTDS is an upper body motion capture suit integrating a number of IMUs that are attached to the upper body using elastic straps.

The MTDS multiplexer is a small belt-worn device that powers the suit from four AA batteries. The device functions primarily wirelessly as it uses a Bluetooth emitter to send data to the computer. While a serial port connector can be used, the wired version is primarily used to update the microcontroller firmware. The multiplexer is the central node of the BSN and its purpose is to acquire, validate, package and send data from each sensor.

The MTDS IMU is a thumb-sized device containing a gyroscope, an accelerometer and a microcontroller. This configuration demonstrates the integration of multiple motion sensors within the same IMU. While a magnetometer would be desirable to for sensor fusion, the magnetometer was not integrated in this iteration of the system. In addition to cost and development time constraints, a magnetometer was not integrated because the scope of this research did not include the development of a new sensor fusion algorithm. The extended Kalman filter is a well-known and highly-accurate solution for gyroscope, accelerometer and magnetometer sensor fusion (see Section 2.5.4). The performance of the MTDS IMU relies on the gyroscope. The IMU's microcontroller is used to extract sensor data and convert it into world-space quaternions as required for the Skeletrix software environment.

4.2 Framework Relevance

The developments presented in this chapter are important in demonstrating the functionality of the Skeletrix software environment. As previously mentioned, the functionality is demonstrated by integrating the two IMUs that are available commercially. The knowledge acquired from the two IMUs is then used to develop the MTDS prototype. The integration and development of the sensors is achieved using the Skeletrix software environment driver development kit (DDK) and BVHE features to form a gateway for inputting inertial motion data into the framework.

As shown in Figure 4-2, the process of integrating hardware is an important layer of the Skeletrix framework. It provides the framework with motion data that can later be used to demonstrate its functionality.

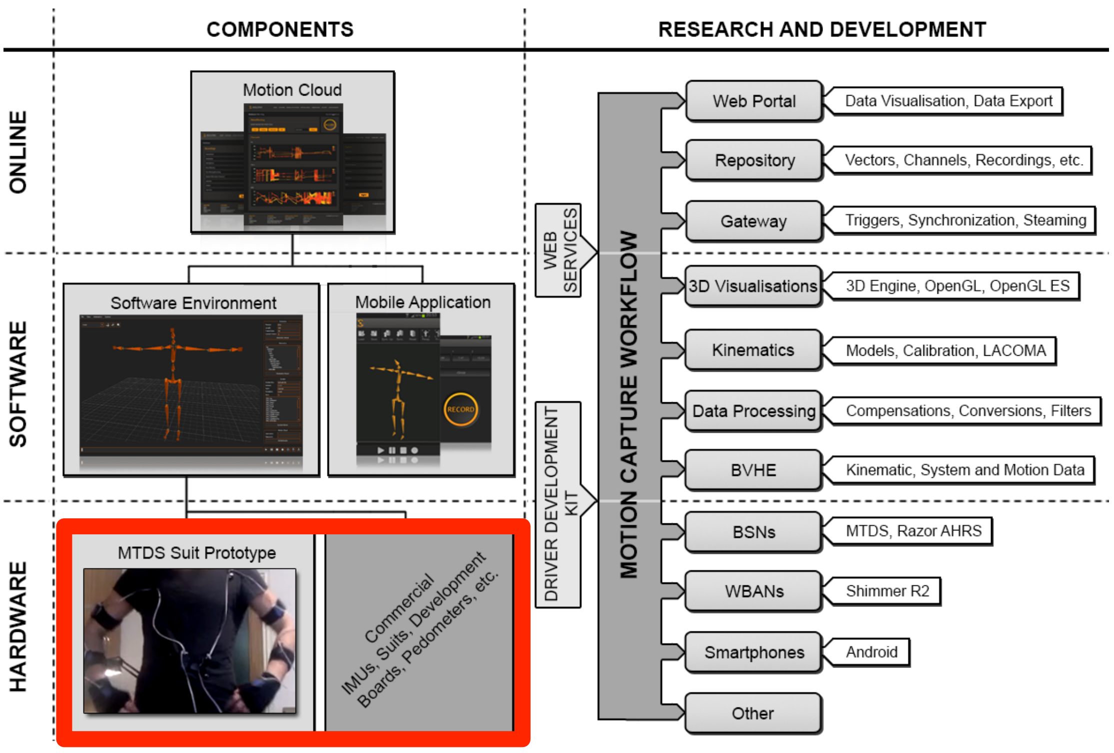


Figure 4-2: The Skeletrix software environment is evaluated using commercial IMUs and MTDS, a prototype BSN. **Source: Pascu et al. [15] [16]**

4.3 Integrating Hardware

This section is focused on the integration of inertial motion capture sensors that are available commercially. Drivers are developed for two very different sensors to demonstrate the integration of a wireless software-centric and a wired hardware-centric IMU.

4.3.1 Case Study: Shimmer R2

The Shimmer R2 [144], shown in Figure 4-3, is a wireless IMU that can be bought individually or as part of the Shimmer software development kit (SDK). The SDK includes documentation, software and firmware examples. The device contains a Texas Instruments MSP430 microcontroller [145] and an InvenSense IDG500 gyroscope [146], an ADXL345 [147] accelerometer and a Honeywell HMC5843 [148] magnetometer. The gyroscope is very similar to the one in the Nintendo Wii Motion Plus controller [149].

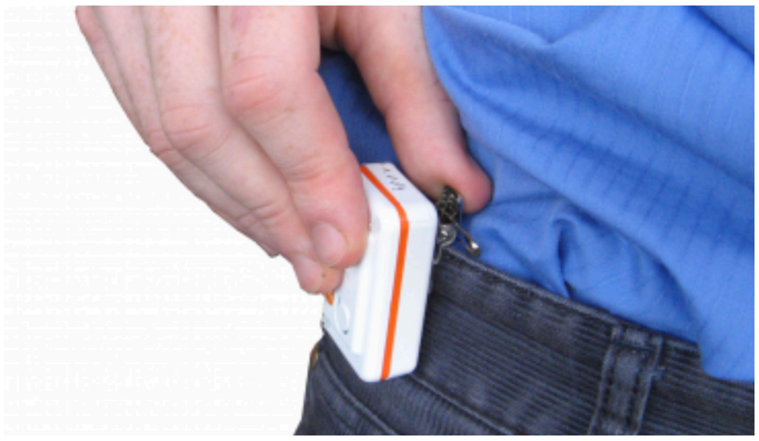


Figure 4-3: Shimmer R2 wireless IMU. **Source: [150]**

Data can be extracted from the Shimmer R2 either through Bluetooth or 802.15.4 radio. Bluetooth was chosen because it is compatible with most laptop computers and does not require an additional receiver. The integration of these devices, to form a basic wireless body area network (WBAN), has previously been discussed in Pascu et al. [19]. Each IMU is paired with the computer through a virtual communication port (VCP) service using a predefined password that is written on the device itself. VCPs are ideal for development because they emulate serial communication ports. In essence, the computer can access the device by opening and querying a port. The microcontroller can be programmed with testing firmware to determine if the device is communicating with the computer correctly. An instruction can be sent to the device to toggle its light-emitting diode (LED), giving the user a visual confirmation that the connection to the computer is successful.

Once connected, the device will not start streaming data without a handshake procedure. The handshake implies sending a set of instructions to activate data streaming. That list of instruction is available within the device's documentation. For devices that do not come with an SDK, a serial port listener can be used to observe, record and replicate how the hardware communicates with its factory software. The handshake instructions were replicated in a driver module, which was developed according to the DDK specifications discussed in Chapter 3.

Once the Skeletrix software environment is connected and data is being streamed, focus is put on understanding the structure of the incoming messages. Data is received as a list of bytes as the device outputs gyroscope, accelerometer and magnetometer readings along with a timestamp and a byte delimiter. Because data is packaged as a stream of bytes, the delimiter allows the driver to understand where a sensor reading starts and finishes. A small parser can be developed to decode the data. The three sensors output raw data that requires conversions. For example, the gyroscope outputs angular speeds that must be converted into world-space rotations.

The Shimmer R2 is a wireless IMU that is suitable for experimental research and development as it provides a first-hand understanding of how an IMU works through tutorials and example source code. However, a BSN integrating a large number of these devices cannot be constructed due to Bluetooth limitations. This problem becomes apparent in a research laboratory where there are a many wireless devices.

4.3.2 Case Study: Razor AHRS

The Razor Attitude Heading Reference System (AHRS) [151] is a wired IMU designed to be integrated like an Arduino development board. As shown in Figure 4-4, this IMU was integrated using a Future Technology Devices International (FTDI) basic breakout board [152] and six connecting cables. The breakout board is used to convert and output the IMU's data through USB. The IMU contains an ITG3200 gyroscope [153], an ADXL345 accelerometer [147] and a HMC5883L magnetometer [154], making it a 9 degrees of freedom (DOF) IMU. Its microcontroller is programmed with an STK500V1 boot loader. Therefore, a computer will recognise this device as being an Arduino development board.

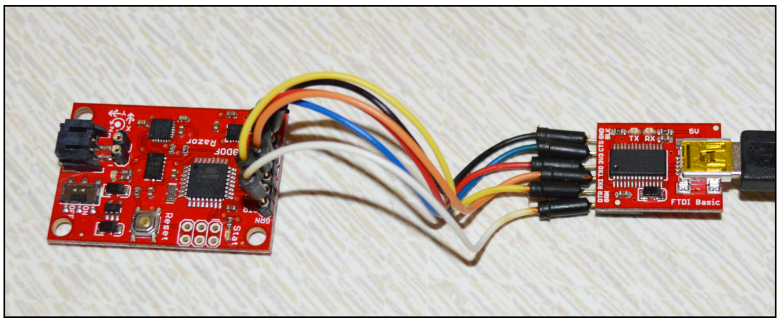


Figure 4-4: Razor AHRS setup with a FTDI basic breakout board.

The device will begin streaming rotational data without any additional instructions or handshaking procedures. The microcontroller is programmed to automatically combine the data produced by the three sensors to produce world-space Euler rotations as required for most application areas. This IMU was integrated with the Skeletrix software environment by converting those Euler rotations into quaternions. The driver was developed in a similar fashion to the Shimmer R2 driver. The source code for the driver can be found in Appendix B. As shown in Figure 4-5, the rotations are applied to the skeleton's root to rotate it in 3D space. The skeleton replicates the rotation of the device as it is tilted.

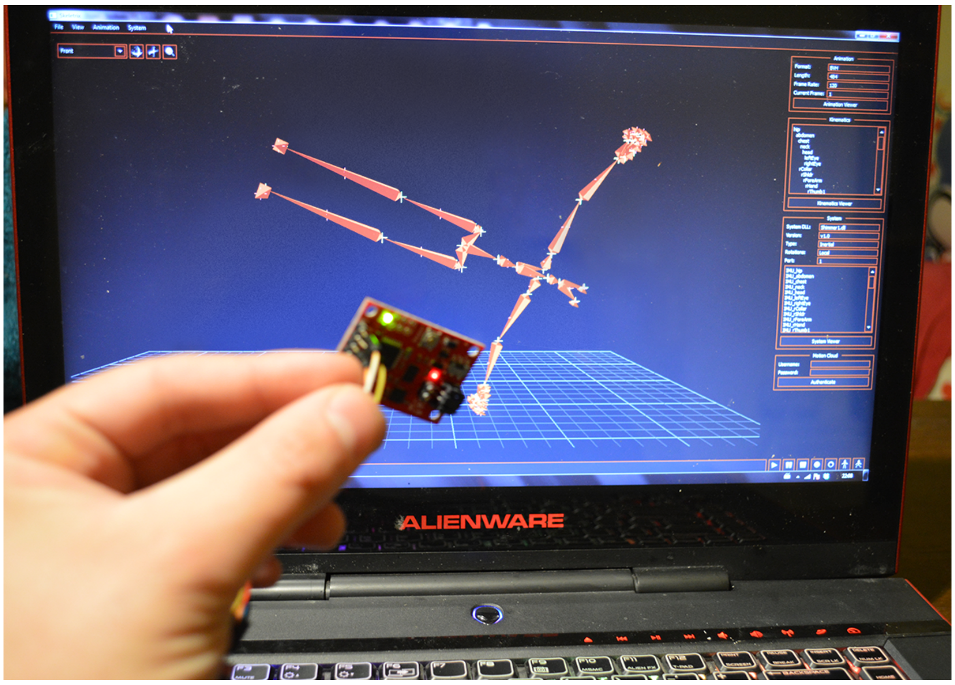


Figure 4-5: Acquiring data from the Razor AHRS.

4.4 Motion Tracking Development System

The Motion Tracking Development System (MTDS) is a purpose-built inertial motion capture suit designed to be worn on the upper body. It is developed in conjunction with the Skeletrix software environment (Pascu et al. [17] [18]) presented Chapter 3. While there are many IMUs available commercially, most devices are not designed to be interconnected to form a BSN. The MTDS contains several low-power IMU enclosing gyroscopes and accelerometers. Unlike other sensor products, the MTDS IMUs are specifically designed to be interconnected and form a BSN. The connection is achieved using a central multiplexer that is tasked with acquiring data from the sensors, packaging it and sending the package to the computer. This section discusses the requirements specification, conceptualization, design and implementation of the suit.

4.4.1 Preliminary Requirements Specification

The MTDS system is a BSN aimed at upper body character motion tracking. As previously discussed in Section 2.6, the development of any BSN should be achieved in accordance to four properties: hardware/software-centricity of the data processing, homogenous/heterogeneous nature of the sensors, directionality of the data intercommunications and networking. This section discusses those properties in the context of MTDS to produce a preliminary requirements specification.

Hardware/Software-Centricity: Inertial motion capture systems require motion processing whereby the sensor outputs are converted in a suitable format that can be used to animate a kinematic skeleton. Motion processing can take place either at the hardware or software level. Software-centricity puts focus on computer drivers while hardware-centricity puts focus on firmware. Like the Razor AHRS, MTDS will compute motion within the hardware to reduce the size of the data packets communicated between the hardware and the computer. A hardware-centric BSN will be easier to integrate with software (thinner software layer as the computations are achieved in the firmware) and perform better (firmware can process motion data faster than software because it is closer to the its source). These benefits, justifying the need for hardware-centricity, are also discussed in Pascu et al. [17].

Heterogeneous/Homogenous: Prototyping a system implies printing circuit boards, manufacturing hardware and developing firmware. The complexity and cost of the system can be reduced considerably by creating one sensor and replicating it throughout the BSN. Therefore, MTDS should be homogenous system.

Directionality: MTDS must be a directional BSN whereby data is extracted from the sensors by the multiplexer and sent, directionally, directly to the computer as quickly as possible. Because the kinematic model is processed on the computer by the Skeletrix software environment, there is no logical reason for omnidirectional node communications in this BSN because it is hardware-centric and no data is sent from the software to the IMUs. An omnidirectional BSN is introduced in Chapter 5 in the context of mobile computing technologies and its benefits are also discussed.

Networking: Data communications between BSN nodes can be achieved wirelessly, using wires or a combination of the two. As previously in the Shimmer R2 case study, adding wireless connectivity to every node in the network can be problematic due to the limited number of channels of supported by Bluetooth. Consequently, node intercommunications must be achieved using a lightweight cable that is both elastic and robust. However, it is desirable that all computer-hardware intercommunications are achieved wirelessly through Bluetooth to ensure that cables do not restrict movement. This approach can be found in the Animazoo [21] and XSens [98] [99] suits.

Motion Processing: The integration of an MEMS gyroscope and accelerometer will require motion processing. Consequently, it is important to understand what the sensors output. The question arises: how usable is the motion data in the context of skeletal motion reconstruction? Like the Shimmer R2, this gyroscope outputs angular speeds that require conversions. Therefore, a small microcontroller must be implemented to perform the conversions. As required for the Skeletrix software environment, rotational data in the form of angular speeds must be converted into world-space quaternions. The quaternions must be compensated to reduce drift and to calibrate the sensors. The notion of calibration has previously been discussed in Section 2.5.2.

Cost: Like with any hardware development, cost plays a very important role in the design and implementation process of a system. Aside from development costs, cost also concerns the target audience. For example, a complex system containing military grade gyroscopes is expected to produce highly accurate motion data. However, MTDS is a prototype system using MEMS sensor chips that are available commercially. It investigates whether those sensor chips are sufficiently accurate to detect the articulated movement of the upper body.

4.4.2 Conceptualizing a Motion Capture Suit

Figure 4-6 illustrates how the MTDS suit works by highlighting its principal components and how they communicate. The computer, running the Skeletrix software environment, communicates with the suit through the MTDS multiplexer. The multiplexer encloses a microcontroller, a battery pack and a Bluetooth module. The multiplexer communicates with the seven IMUs through a harness containing only three cables. The first cable (red) must be used to send data (e.g. firmware updates, polling commands, instructions, etc.) to the BSN nodes. The second cable (green) is used to retrieve data such as angular readings. Lastly, the third cable (blue) powers the whole system from the multiplexer's battery pack.

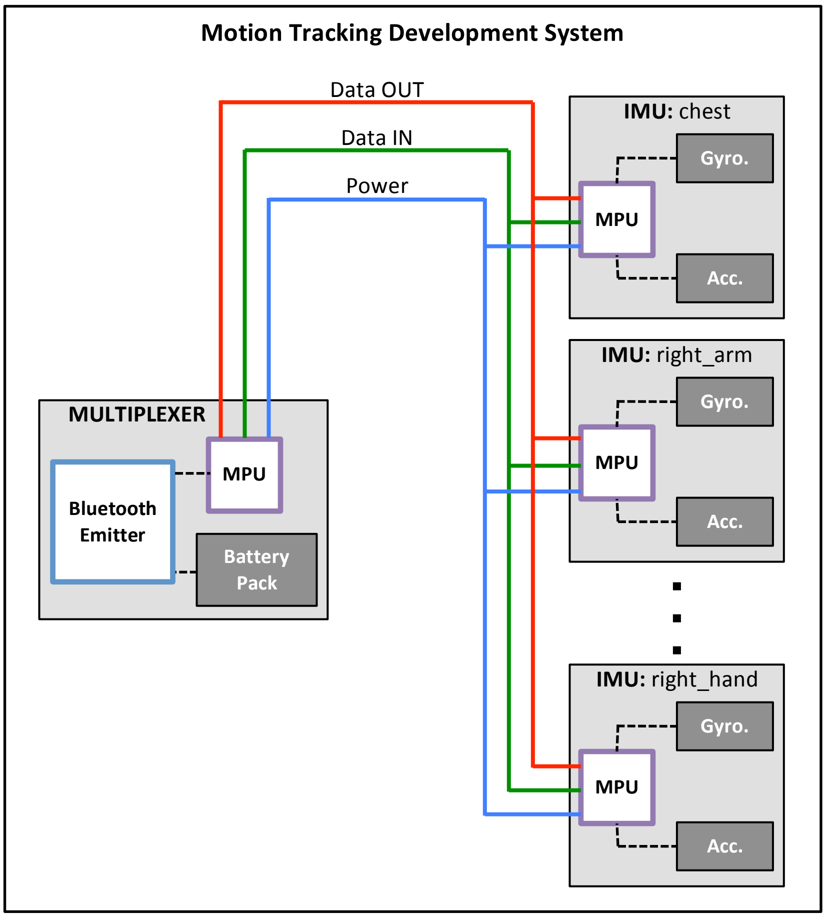


Figure 4-6: How seven MTDS IMUs communicate motion data to the multiplexer.

As shown in Figure 4-7, the MTDS suit uses seven IMUs placed on the hands, forearms, arms and torso. The suit consists of seven straps, which can be worn on top of clothing. More specifically, the hands use fingerless gloves, the forearms use modified wristbands, the arms and torso use elastic straps. Two additional straps are wrapped around the shoulders to prevent the harness from getting in the way of the arm movements.

The IMUs are daisy chained using a harness, which begins at the multiplexer, bifurcates at the chest and finished at the hands. The initial prototype used a ribbon cable but, due to its length, the three signals were interfering with each other. Additionally, the ribbon cable proved brittle and often broke at the connectors. The solution was to take three slightly thicker cables and plat them. The platted harness proved robust and elastic, making it less likely to break at the connectors.

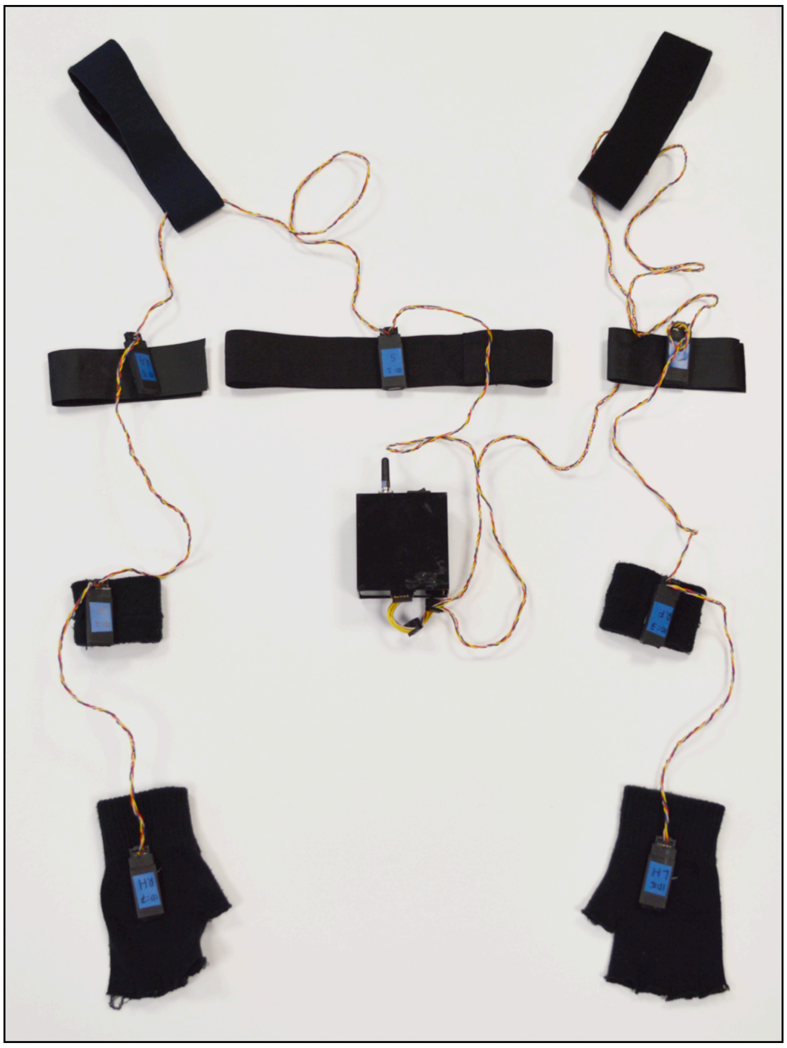


Figure 4-7: MTDS suit.

4.4.3 MTDS Inertial Measurement Unit

The MTDS IMU is a small thumb-sized device designed to measure either rotations or gravitational accelerations of a body part. The device was designed to be lightweight so that it moves very little in relation to the body. The device was also designed to be flat so that it can be attached to the body with elastic straps. As shown in Figure 4-8, the device is a small circuit board with an insulation-displacement connector (IDC). The internal components are protected by heat-shrink rubber tube.

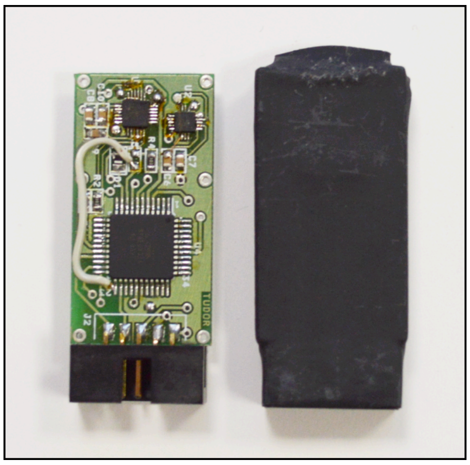


Figure 4-8: MTDS inertial measurement unit.

The circuit board integrates IMU3000 [155] gyroscope and, depending on availability, both MMA8451Q and MMA8452Q accelerometers [156]. At the time of development, the IMU3000 was one of the most powerful consumer-level gyroscopes. It is a newer and more powerful version of the IDG500, which can be found in Shimmer R2 IMU. The microcontroller used to process the motion is a low-power Atmel AVR RISC chip [157].

The desired output of the device is world-space quaternion rotations, as required by the Skeletrix software environment. Like the Shimmer R2, the IMU3000 outputs angular speeds instead of the more desirable world-space rotations. Consequently, the MTDS IMU's microcontroller firmware had to convert angular speeds into quaternions. This conversion is a three-step process. To begin with, the microcontroller stores a timestamp with every recorded angular speed. Rotational increments can be calculated by multiplying the angular speed with the timestamp difference. Then, rotational increments are summed to produce world-space Euler rotations. Lastly, the rotations are converted into quaternions using the conversion equations shown in Appendix A.

The MTDS IMU has two operational modes that can be selected through a boot loader. On start-up, the boot loader decides which part of the firmware code to execute. The default operational mode is the development mode, which allows for firmware updates from the computer. Alternatively, the device will function as a BSN node whereby it outputs quaternions if a specific instruction is sent to the microcontroller.

The MTDS IMU was developed to demonstrate the construction of a BSN and data acquisition from multiple motion sensors. A magnetometer was not necessary for this task. However, without a magnetometer sensor fusion is not possible. Alternative methods for drift compensation were used to reduce the amount of gyroscope drift. When started, the device must remain motionless for 10 seconds. Even though the device is motionless, the gyroscope outputs rotations that correspond to gyroscope drift. The microcontroller starts to read and sum those rotations and, after 10 seconds, concludes a compensation value. The compensation value is subtracted from every gyroscope output to follow, thus calibrating the gyroscope for use. Using this approach means that the sensor can only be used for short periods of time before drift becomes noticeable. Drift could be further compensated by integrating a magnetometer and performing sensor fusion. Notably, modern motion sensors (e.g. the InvenSense MPU9150 [158]) contain the gyroscope, accelerometer and magnetometer in the same chip capsule and perform sensor fusion automatically using an internal microcontroller. Such devices could be integrated in future iterations of the MTDS IMU to solve the drift problem.

4.4.4 MTDS Multiplexer

The MTDS multiplexer is the central node and power supply of the BSN. Because it contains four AA batteries, it is also the heaviest component of the suit. As shown in Figure 4-9, the small metal box is worn using a belt hook. Because the multiplexer is the main component of the suit, it contains the power switch that turns on and off all the IMUs.

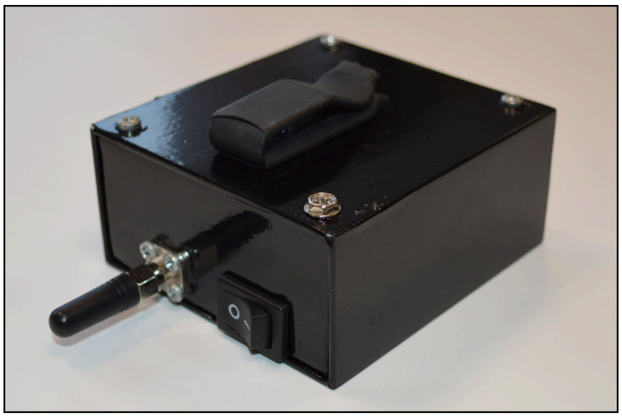


Figure 4-9: MTDS multiplexer.

As illustrated in Figure 4-10, the multiplexer contains a Bluetooth module, an IDC connector for the suit's harness and two twin AA battery packs that power the suit. The multiplexer also encloses a MTDS IMU, which is integrated as part of the main circuit board. This means that there is a gyroscope and an accelerometer inside the device. However, the multiplexer's sensors were made redundant for three reasons. First, the device's microcontroller is not powerful enough to simultaneously process motion and acquire data from the BSN. Second, the multiplexer proved too heavy to be worn on the chest and wearing it on the belt gives inaccurate motion readings. Third, it proved time consuming to develop two separate iterations of the IMU firmware, one for the BSN nodes and one for the multiplexer's sensors. The solution to these problems was to add an additional IMU to the BSN, which is external and connected to the multiplexer using the harness.

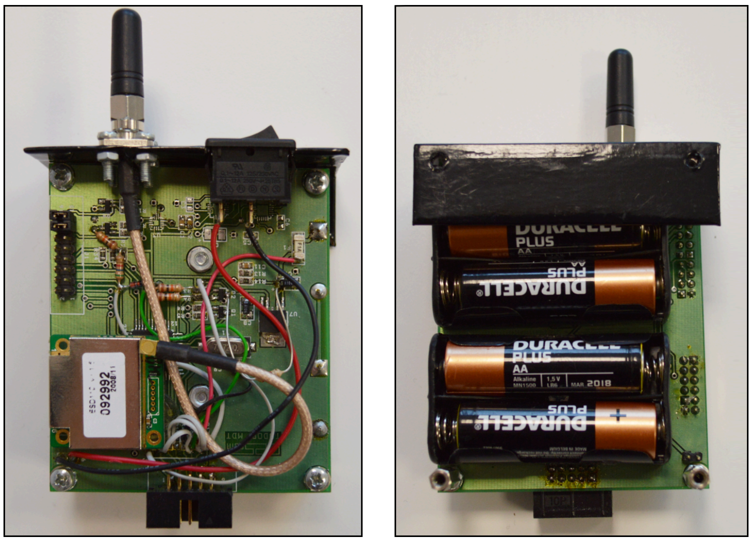


Figure 4-10: MTDS multiplexer opened showing top (left) and bottom (right) views.

The multiplexer has three methods for communicating with the computer. The suit can be connected to the computer using a serial connector or through a USB convertor. The wired approach is primarily aimed at developing or updating or firmware for the multiplexer and the network's IMUs. Alternatively, the suit can be used wirelessly though Bluetooth. As shown in Figure 4-4, the multiplexer integrates a Parani ESD100 Bluetooth module [159]. This module takes the input of a serial cable and outputs a Bluetooth signal that can be interpreted by the computer as a serial connection through a virtual port emulator. While the wireless approach is more desirable, the wired approach is more stable. In both scenarios, the suit is powered by the four AA batteries and not through the USB.

The multiplexer's microcontroller is an Atmel AVR RISC chip that is similar to the ones implemented in the IMUs. The microcontroller polls the sensors at specific time intervals while taking into account code execution delays to ensure that data is collected at the exactly the specified frame rate. The microcontroller does not have sufficient Random Access Memory (RAM) and there is no internal storage to buffer the result for longer periods of time, especially at high frame rates. Every time an IMU is polled, the result is sent straight to the Bluetooth module that forwards it to the computer. The IMU polling takes place in a sequential fashion. The multiplexer iterates through the network to poll every IMU individually. Data is extracted and packaged along with a unique identifier, which keeps track of which sensor has produced the reading.

The multiplexer verifies the validity of the data using a cyclic redundancy check (CRC) function to ensure that the incoming sensor readings are feasible. The CRC function uses hash encoding and compares every sensor's reading against previous readings. An incomplete reading or error will be discarded and replaced by a previous reading. If there is a problem (e.g. a sensor disconnects), this safety measure ensures that the BSN does not crash. Instead, the BSN remains partially operational until a sensor is reconnected.

4.4.5 Upper Body Character Motion Tracking

The MTDS suit was evaluated through a series of upper body motion tracking experiments where the motion performer wears the suit and acts out upper body gestures. As shown in Figure 4-11, the kinematic model replicates the posture and motion of the performer real-time at the frame rate of 30f/s.

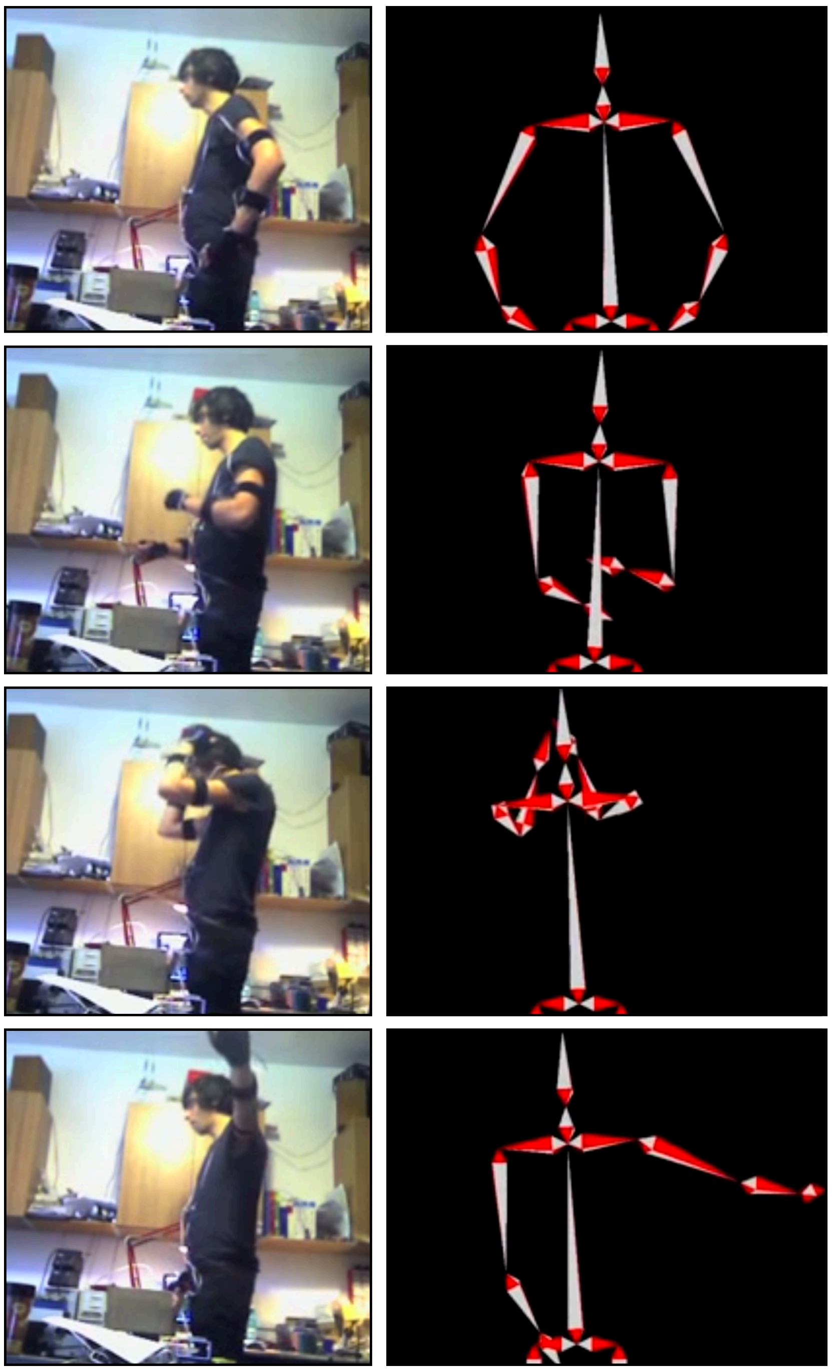


Figure 4-11: MTDS upper body motion tracking using the Skeletrix software environment.

Source: Pascu et al. [17]

4.5 Conclusions

This chapter has demonstrated the integration of existing sensor hardware within the framework using the Skeletrix software environment. This chapter has two fundamental goals. The first goal is to demonstrate the integration of hardware, which is an important requirement for the framework. Hardware integration is required to create a source for inertial motion capture data that can later be used to test and evaluate the framework. For example, data recorded from these studies can be uploaded to the Motion Cloud and visualised using the web portal or the smartphone application presented in Chapter 5. The second goal is to discuss and demonstrate BSN development, a task which is achieved through the Shimmer R2, Razor AHRS and MTDS prototype.

MTDS demonstrates how a BSN prototype can be constructed cost-effectively to be robust and suitable for experiments (due to the addition of a boot-loader that facilitates firmware updates), as required for the research work presented in this thesis. Its development is presented in three stages. The initial stage is focused on the conceptualization and requirement specification of the system to establish how the system will work in terms of hardware-centricity, connectivity, power consumption, etc. The second stage showcases the development of a multiplexer device, which is a central node for the network. The third stage is focused on the development and implementation of an IMU that is designed specifically to function as a BSN component.

The MTDS suit was not benchmarked against other systems because there are no motion capture suits that provide SDKs or open source code. Benchmarking is a process that requires the testing and evaluation of each stage of the motion capture workflow (e.g. comparing methods for calibration, sensor fusion, etc.). Motion capture suit manufacturers, such as Animazoo [21] or XSens [98] [99], tend to protect their intellectual property and do not provide users with the source code. Instead of benchmarking, the three IMUs were compared and evaluated in accordance to the sensor attributes previously established in Chapter 2 in order to highlight the strengths and weaknesses of each sensor within the context of constructing a BSN.

The Shimmer R2 IMU is the least powerful of the three presented and also the oldest. It is the only device that provides wireless connectivity and contains a battery. Using the SDK, it is also the device most suitable for experimental research. Unlike the MTDS IMU, it includes a magnetometer and has potential to be flashed with sensor fusion firmware. Unlike the Razor AHRS, it comes in a small package that can be strapped to the body. Shimmer even supplies a strap with the device. Due to the limitations of Bluetooth many devices cannot be networked to form large WBANs.

The Razor AHRS IMU is the most powerful of the three presented and also the newest. It is the only device that runs a sensor fusion algorithm. Consequently, it presents the least amount of gyroscope drift and can be used for long periods of time. The device is capable of producing world-space rotations straight out of the box. This device contains the most powerful gyroscope, accelerometer and magnetometer out of the three presented as its technical specifications are closely matched to the MTDS IMU. Like the MTDS IMU, this device can be used to construct a BSN by replacing the FTDI breakout board with a multiplexer.

By today's standards, the MTDS IMU contains a mid-range gyroscope and accelerometer. Considering its preliminary requirements specification, this device is the smallest of the three in terms of size and is ideal for constructing a BSN. Motion is processed at the hardware level with the aid of the multiplexer.

These three sensors can be used to showcase the advances in MEMS technologies that took place throughout the completion of this thesis. Modern sensors, like the Razor AHRS, come programmed with sensor fusion algorithms from the factory. The successor to the Shimmer R2, namely the Shimmer 3 [160], also provides that functionality. Sensor fusion is one of the most problematic yet important properties of MEMS technologies.

CHAPTER FIVE

5 Sensing Through Mobile Computing Technologies

5.1 Introduction

Chapter 3 has introduced the Skeletrix software environment published in Pascu et al. [17] [18], a tool for developing inertial motion capture technologies. Emphasis was put on the individual challenges faced in extracting data from hardware and producing 3D visualisations. As a result, this thesis has established a more efficient motion capture workflow aimed at tightening the relationship between hardware and software. Using the driver development kit, users can construct and integrate body sensor networks (BSN) using heterogeneous sensors. The hardware configuration and its output are stored in a revised file format entitled Biovision Hierarchy Extended (BVHE). To summarise, the software environment has presented a white-box approach to creating a motion capture workflow through which all the individual computations of motion are exposed to the developer.

The smartphone is the most ubiquitous [161] wearable computing technology and its sensing capabilities have many application areas [162] [163]. Most people have smartphones and most smartphones enclose gyroscopes, accelerometers and magnetometers. This configuration is identical to that found in inertial measurement units (IMU). The smartphone's worldwide uptake has prompted many advances in MEMS technologies whereby inertial motion sensors are now designed more robustly and are more affordable. In addition to inertial sensors, modern smartphones also enclose global positioning systems, optical cameras, pressure sensors, thermometers and light sensors. The smartphone's inherent capability to sense while connected to the Internet has produced new application areas. When paired with web technologies, modern smartphones can form large sensor networks [164] focused around aggregating sensor data in online repositories. A popular application area for smartphone sensor networks is activity tracking whereby the device is able to determine the overall comportment of its operator [165] [166]. This chapter poses three important questions: What is the fundamental difference between sensor networks and BSNs? Is it possible to construct BSNs using smartphones? If yes, is human motion capture possible through smartphone-driven BSNs?

To answer those questions, this chapter introduces an innovative mobile application, which was published in Pascu et al. [15] [16] [19]. The mobile application makes use of the smartphone's ability to constantly sustain an Internet connection to establish online BSNs. The goal is to create a motion capture system that can sense, compute and visualise motion completely independently of a computer. Using the mobile application, several smartphones running instances of the application can be interconnected over Wi-Fi or 3G. While a typical motion capture suit uses a multiplexer as a data hub between its constituent nodes, the proposed smartphone equivalent substitutes the concept of a multiplexer for an online server, which is entitled Motion Cloud and is further discussed in Chapter 6. Each device computes motion and uploads it in real-time to a server through a set of web services. The server merges the result and directs it back to every smartphone in the network, thus forming an omnidirectional BSN. The unique property of omnidirectional networks is that every sensor node communicates with every other node, thus eliminating the need for ranked roles (there are no master-slave relationships between network components). Every node is aware what every other node's data, allowing it to compute motion more accurately. Additionally, the workload is distributed between several multicore processors making the BSN computations more efficient.

Aside from the novel approach to capturing motion, the Skeletrix mobile application is unique in several aspects. While most inertial suits are developed to function in one particular configuration, the online server dictates the configuration of the smartphone-driven BSN, making the system highly customizable and modular. The user has the option to tailor the system to every individual experiment by choosing the number of smartphones to interconnect. This approach simplifies the task of adding or subtracting smartphones from a BSN. The user also has the option of creating very large BSNs because the server can support more connections than a physical multiplexer.

In contrast to the software environment previously introduced, this chapter presents an automated workflow in accordance to the black-box paradigm whereby the software environment processes motion without user input. Like the software environment, the smartphone application also contains kinematic models, an OpenGL ES renderer, file parsers, etc. Condensing the otherwise large components of the software environment into a mobile application required much optimization and simplification both at the view and controller layers. The mobile application is developed as a Model View Controller (MVC) architecture.

To put the development of the mobile application into context, three fundamental questions arise. What could the mobile application be used for? What are its application areas? Are people going to strap smartphones to their bodies? The mobile application is not presented as a substitute for inertial motion capture systems, but as a suitable test bed for prototyping BSNs and creating small systems for experimental research. In situations where the experiment requires two or three sensors, it may be simpler to use smartphones than the much more difficult to attain and expensive motion capture alternatives.

Users sometimes do wear smartphones strapped to their bodies, particularly in the context of activity tracking. For example, there are armbands available commercially that allow users to attach the smartphone to the arm while jogging to measure their physical activity in terms of number of steps, distance travelled, etc. Medical science is another very important application area for smartphones. Some medical disorders, such as idiopathic scoliosis [51] or Parkinson’s disease [48] [79], have a measurable effect on the motor functions of the body. The smartphone application could be used to track and evaluate the behaviour of patients. Because the smartphone application streams data in real-time, it could also be used to find anomalies that are indicative of emergency situations (e.g. an elderly person falling). These and other application areas are further discussed in Section 5.13.1.

5.2 Framework Relevance

This thesis investigates new technologies that may be relevant to inertial motion capture and BSNs. The mobile application explores the concept of sensing through mobile computing technologies. Although the mobile application has a unique purpose and functionality within the proposed framework, its development is fundamentally based on the Skeletrix software environment architecture. More specifically, the Skeletrix software environment is used as a starting point for the development of the mobile application. The revisited workflow is designed to take into account the computational constraints of smartphones while following the black-box paradigm to create a user-friendly system.

While inertial motion capture is an important research topic that is relevant to many application areas (see Section 2.3), inertial BSNs are a niche sector of the animation and biomechanics industries. Because the smartphone is ubiquitous, the addition of mobile computing technologies to the Skeletrix framework creates a new spectrum of application areas centred on sensor networks. The mobile application is also a new source for motion data within the framework. One of the biggest contributions of the mobile application to the Skeletrix framework is the addition of web technologies to the workflow. Although the Motion Cloud is introduced in Chapter 6, its conceptualization originates from this chapter's developments. Figure 5-1 illustrates this research work's place in the framework.

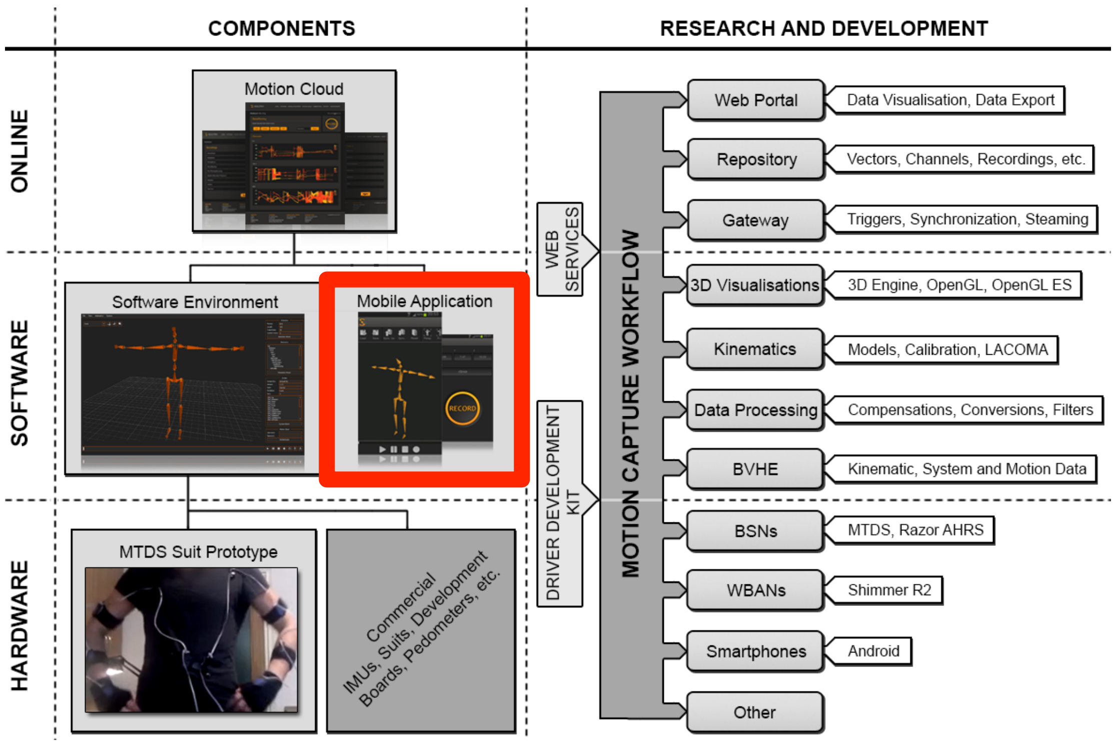


Figure 5-1: Skeletrix mobile application introduces web and mobile computing technologies to the framework. **Source: Pascu et al. [15] [16]**

5.3 Preliminary Requirements Specification

As previously mentioned, the development of the online smartphone-driven BSN is derived from the Skeletrix software environment presented in Chapter 3. The process of porting the software environment to the Android platform involves a major redesign of the architecture in terms of front-end and back-end functionality. This section summarizes both functional and non-functional requirements to create a preliminary requirement specification. The resulting specification can be divided into system, interface and network requirements.

5.3.1 System Requirements

Object-Orientation: The Android platform uses Java, which is an object-oriented programming language that runs in a virtual machine. Object orientation allows the code to be efficient, modular, reusable and easy to maintain or extend. Consequently, the Java-based mobile application must be object-oriented whereby the code is divided into object and each object has a logical purpose by itself and as a component in the architecture. While memory management and garbage collection is achieved through the virtual machine, efficient object intercommunications are key in optimizing performance.

Multithreading: Modern mobile computing technologies enclose multicore ARM or Intel processors that support multithreading. The smartphone requires a powerful processing unit to run multiple applications concurrently. Inertial motion capture implies computing motion as quickly as possible and multithreading is an integral part of that process. The tasks of processing motion data, computing kinematics, rendering 3D visualisations, running the interface layer need to be allocated separate threads. Multithreading allows the code to execute concurrently on the processor, thus increasing the overall performance.

Distributed Computing: The concept of smartphone-driven BSNs implies using devices that sense and process motion concurrently where a copy of the application is running on every network node. Rather than computing network's motion on one single device, each network node can process its own motion. Distributed computing is key in ensuring an even allocation of tasks across the network. Using a constellation of multicore processors amounts to one computationally powerful system.

Motion Processing: By default, the smartphone does not produce usable motion. Because the accelerometer and magnetometer sensors are used by the interface, the Android Application-Programming Interface API provides functions for extracting and merging the accelerometer/magnetometer orientation. The orientation is provided as a world-space rotation that contains large amounts of noise. Post-processing filters must be applied to make that data usable for motion capture. Alternatively, the gyroscope data can be used as it contains little noise. However, the gyroscope data contains drift that can be compensated using the other two sensors. The ideal result should contain little noise or drift.

5.3.2 Interface Requirements

Black Box Paradigm: While the Skeletrix software environment was focused on creating a level of transparency that helps developers understand motion capture systems, the mobile application must be developed in accordance to the black box model [167] [168]. This is because the act of interacting with several device screens simultaneously is difficult and reducing the number of user-performed action will improve usability. Consequently, the majority of the mobile application's functionality must be autonomous. The black box model defines a type of programming focused on inputs and outputs rather than the intermediary computations. As shown in Figure 5-2, the mobile application must achieve a lot of functionality with little user input from the interface layer.

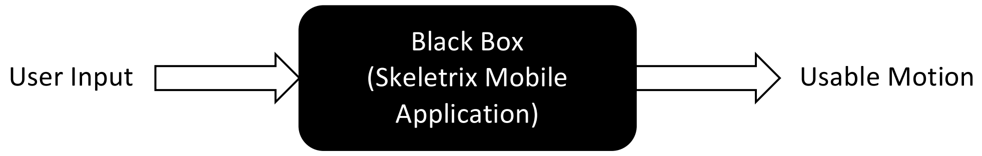


Figure 5-2: The black box model and Skeletrix mobile application.

Multitouch Gesture Interaction: A key design decision for mobile computing technologies that minimise the complexity of user interfaces is multitouch gesture interaction. The study [169], which is primarily focused on tablet interactions, discusses the importance of multitouch gestures in the context of musculoskeletal systems and kinematic models. While the Skeletrix software environment uses mouse and keyboard to interact with the 3D visualisations, the mobile application must replicate that functionality through multitouch gestures. The two most common gestures familiar to smartphone users are pinch to zoom and swipe.

3D Data Visualisation: As previously discussed in the context of the software environment, 3D skeletal representations of motion using kinematic models are important in giving users an interpretable visualisation of motion. Rotational data must be gathered from all BSN nodes and applied to a kinematic model. Because the mobile application targets the Android platform, an OpenGL ES [170] renderer can be used to visualise the motion in 3D as a virtual skeleton.

5.3.3 Network Requirements

Omnidirectional Communications: The task of creating smartphone-driven BSN differs fundamentally from that of an inertial motion capture system. Smartphones present the unique property of combining both the hardware and software aspect of sensing into one device. Consequently, the smartphone-driven BSN does not require a designated computer. This poses the question: which body-worn smartphone becomes the computer? Omnidirectional BSNs are also beneficial because each node can take into consideration the data produced by its neighbouring nodes when computing its own motion, thus making the result more accurate.

BSN Controller: Inertial motion capture systems have multiplexers, which are chest or belt worn devices tasked with gathering data from the sensors, synchronizing the data and sending the combined result to a computer. The multiplexer serves the fundamental purpose of turning several devices into one singular system. The smartphone-driven BSN must replicate that functionality in the absence of a physical multiplexer device. The only solution is to use a server and a set of web services to remote control the functionality of the smartphones and retrieve motion data.

Data Streaming Protocols: While inertial motion capture systems use short-ranged communication protocols (e.g. Bluetooth), the smartphone equivalent will rely on Internet connectivity. Internet connectivity is particularly unstable depending on many factors such as signal strength and connectivity mode (i.e. 3G, 4G, Wi-Fi). A flexible data streaming protocol is required to ensure that no data is lost.

5.4 Conceptualizing a Smartphone-Driven BSN

This section discusses how the smartphone can be converted into a BSN node in one tap. Figure 5-3 illustrates how three smartphones can be interconnected through the Motion Cloud repository and gateway. Each smartphones streams its data into repository channel objects. Simultaneously, each smartphone interrogates the repository to retrieve a complete data set. The BSN controller is a component of the Motion Cloud gateway that controls several BSN nodes remotely. Rather than starting each smartphone manually, the network nodes are connected to a gateway trigger object. Each network node constantly listens for trigger status changes. As a result, users can use one smartphone to control the whole network.

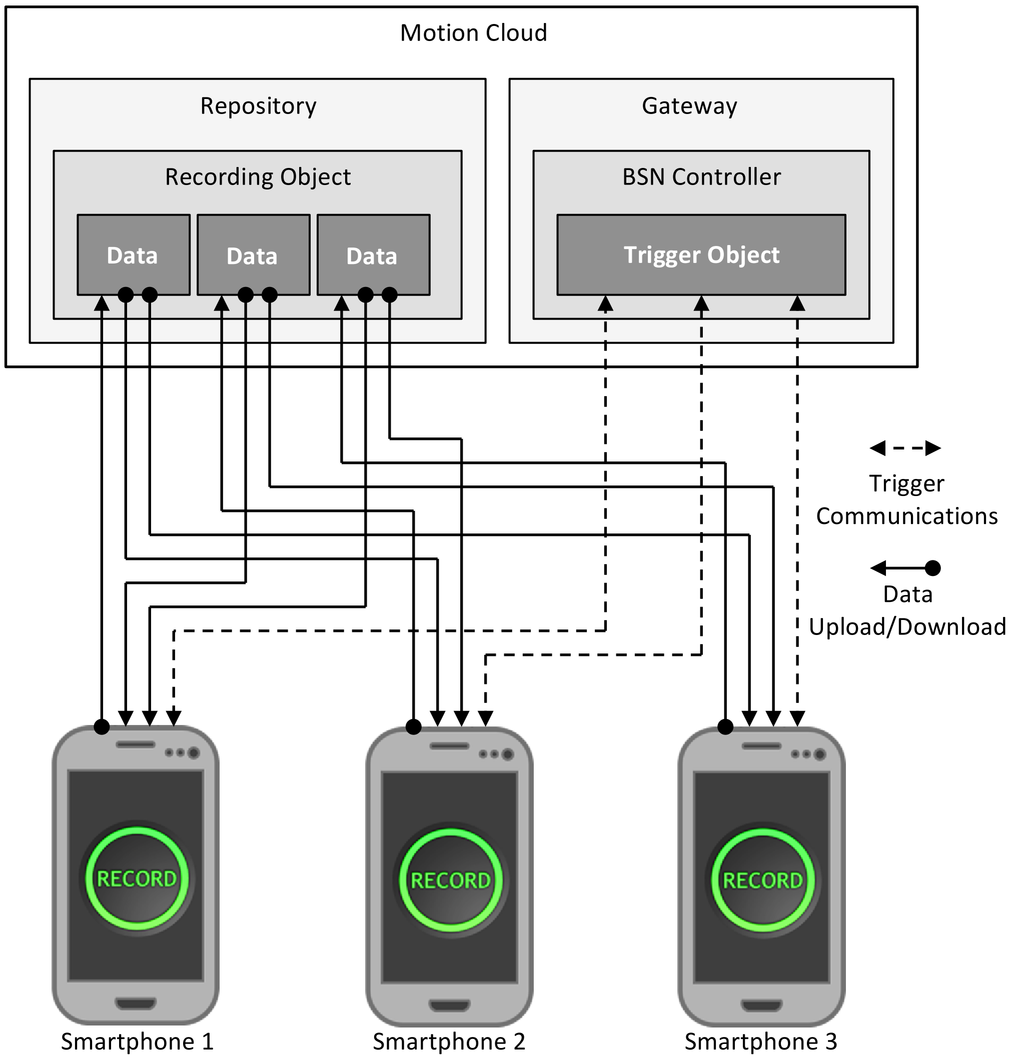


Figure 5-3: Three smartphones form an omnidirectional BSN using the Motion Cloud repository and gateway. **Source: Pascu et al. [15] [16]**

5.4.1 Initializing the BSN

The first stage of initializing the smartphone-driven BSN involves creating a Motion Cloud user account. Once logged in, users can create or select an existing recording object through the Motion Cloud web portal or the mobile application interface. User accounts and recording objects are required to allow multiple BSNs to function simultaneously on the Motion Cloud. Once a recording object is selected, the smartphones become a BSN by listening to a common gateway trigger. A successful BSN initialization can be observed visually as each smartphone's interface turns from orange (inactive) to green (active).

Once the BSN is initiated, the smartphones start streaming data. Figure 5-4 illustrates the internal workings of the mobile application as a flow diagram. The application's recorder extracts and processes the motion data. The application verifies that the smartphone is online and communicates data to the server. The server receives the data and waits for all other nodes to upload. Once a complete set of data is found, the server applies a synchronization algorithm and sends the result back to the smartphones. Each smartphone cleans the data and applies it to the kinematic model, thus producing an animated skeletal rig.

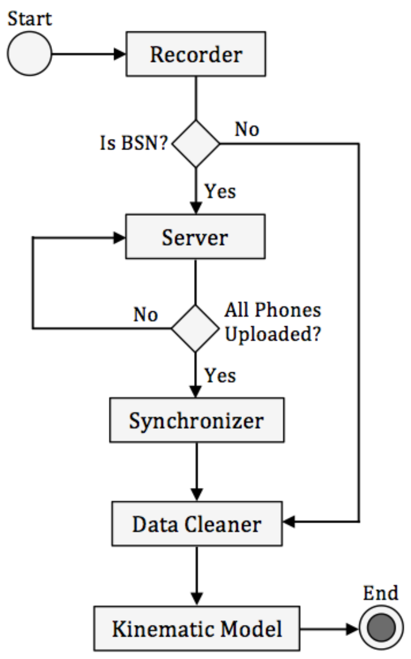


Figure 5-4: The stages of streaming motion data to and from the Motion Cloud server.

Source: Pascu et al. [19]

5.4.2 Operating Modes

The mobile application has been evaluated through four operating modes: offline, online, directional streaming and omnidirectional streaming. Through these operating modes, the proposed smartphone application becomes more versatile in a wider variety of contexts. Directional streaming was implemented in the final version of the mobile application.

Mode 1: Offline

Motion data can be recorded locally without web connectivity. If the smartphone is offline, the server communication stages will be bypassed and the data will be applied directly to the kinematic model. However, the interface allows users to manually upload data to a recording object. As a result, data can be recorded locally and uploaded later.

Mode 2: Online

It is not mandatory that data gets streamed to the server in real-time. The application's second operating mode only uploads data to the server once a recording is complete. Data is buffered locally and the very end of the recording session, each smartphone sends larger packets for server-side synchronization and network distribution. This operating mode is primarily useful for situations in which the wireless signal is weak.

Mode 3: Directional Streaming

The relationship between the server and the smartphone can be configured to be directional whereby smartphones upload data without the server responding until the recording is stopped. The resulting motion can be accessed through the Motion Cloud web portal or at the end of the recording. This operating mode is primarily for situations where the result is only needed at the end of a recording session.

Mode 4: Omnidirectional Streaming

The last and most resource intensive operating mode, showcased in Figure 5-3, involves streaming data to and from the server in real-time. Every smartphone in the network uploads and downloads data from the Motion Cloud as quickly as possible through asynchronous tasks. This operating mode is primarily aimed at Wi-Fi connections.

5.5 Architecture Overview

The Skeletrix mobile application architecture follows the principles of object-oriented programming. The code is written in Java and distributed throughout a set of objects. The objects are grouped into packages whereby a package represents a category of functionalities. The interface is written in eXtensible Markup Language (XML) using restyled Android interface objects. This approach is standard for an Android mobile application. Figure 5-5 shows the architecture's six main packages entitled: launch, core, kinematics, mathematics, sensing and user interface. The diagram also describes objects and object interactions. This section continues to discuss the functionality of each package aside from the user interface.

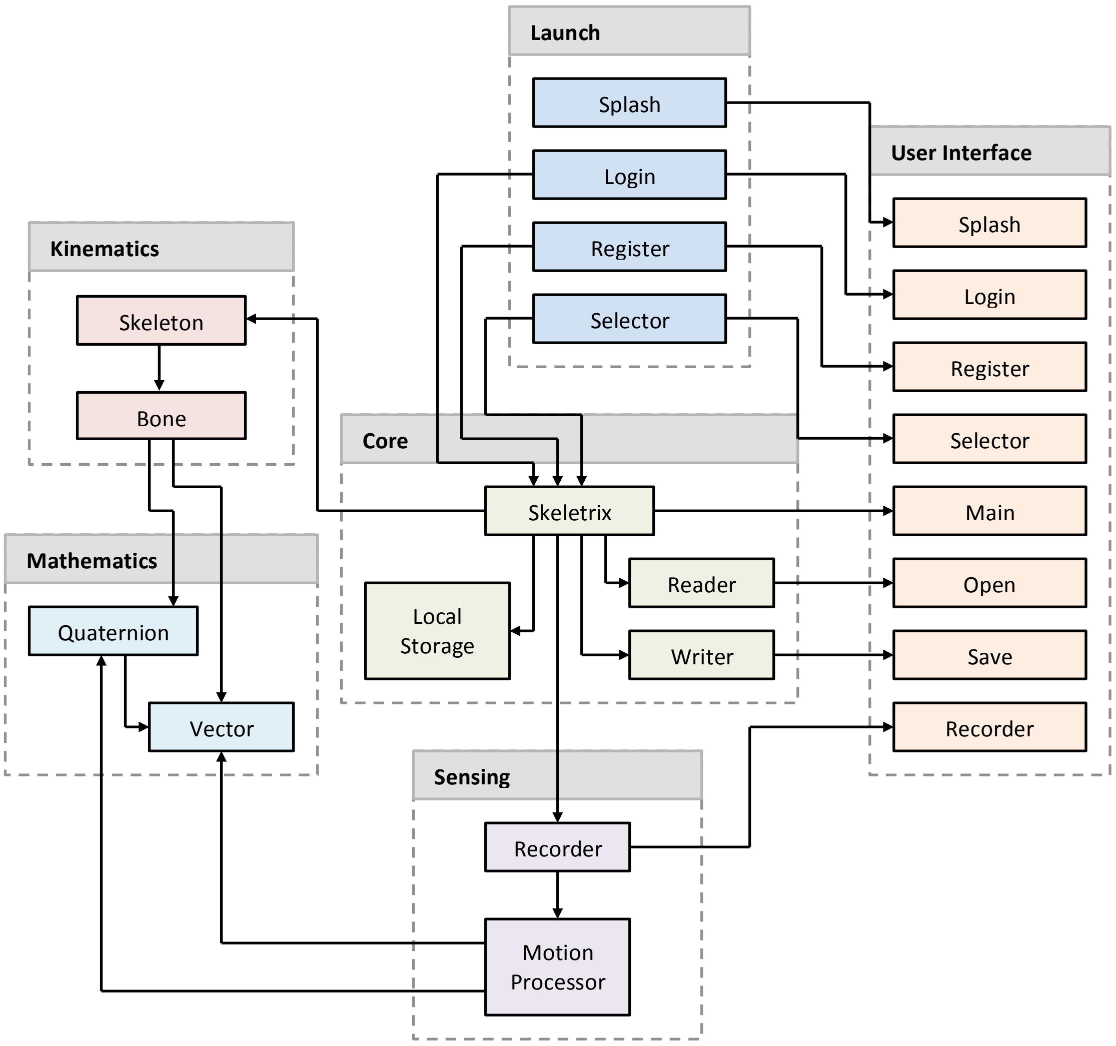


Figure 5-5: Skeletrix mobile application architecture diagram showing the main objects organised as packages.

5.5.1 Launch Package

The launch package provides all the back-end functionality for initiating the mobile application. Its primary focus is to establish a connection with the Motion Cloud that will be used to transfer data between the BSN's nodes.

Login/Register Objects: The login object is used to authenticate the user with the Motion Cloud database. A unique user id is retrieved from the database and passed to the core package. For new users, the register object can be used to create a new Motion Cloud account.

Selector Objects: The selector object is used to select a recording. Recordings define the web space used by the BSN to store motion data. Several smartphones sharing the same recording object automatically become a BSN.

5.5.2 Core Package

As with the software environment, the mobile application contains a centralised core package tasked with gathering and processing the data from every other package. All data passes through the core package at some point.

Skeletrix: The Skeletrix object is the mobile application's main object. It is used to gather and combine data from all the other packages. Other functionalities include 3D rendering kinematic models, multitouch gesture interaction with the viewport and animation controls.

Storage: The storage object is a centralised object tasked with storing, managing and making accessible all data that is shared throughout the application (e.g. user id, recording id, kinematic model).

Reader/Writer Objects: The reader object is used to parse Biovision Hierarchy (BVH) files from the smartphone's Secure Digital (SD) card to create a kinematic model. The writing object is used to output the kinematic model and its motion data as a BVH file. BVHE is not supported because all Android smartphones have one standardised sensor configuration (a single sensor).

5.5.3 Mathematics Package

The mathematics package contains libraries for vector and quaternion algebra and is very similar to that of the Skeletrix software environment.

Vector Object: The vector object provides a library for vector operations, which is used frequently in the context of rotational and positional data. The reader, writer, sensor manager, renderer and interface layer primarily use vectors.

Quaternion Object: For performance reasons and to avoid gimbal lock, quaternions are used to represent rotational transformations. The quaternion object uses some of the functionality of the vector object.

5.5.4 Kinematics Package

The kinematics package generates virtual bone objects and skeletal hierarchies to create kinematic models (or rigs). Those models are subsequently used to create 3D visualisations.

Bone Object: The bone object consists fundamentally of a name, a positional offset and a list of quaternion rotations corresponding to frames of motion.

Skeleton Object: The skeleton object takes a list of bones and forms a skeletal hierarchy consisting of both rotational and positional offsets. The skeleton object provides a list of functions for iterating through the hierarchy. The methodology for iterating through the hierarchy (in a specific order) is important in computing the rotational and positional constraints correctly.

5.5.5 Sensing Package

The sensing package is used to take data from the device's gyroscope, accelerometer and magnetometer, compute a result and send that result throughout the BSN.

Sensor Manager Object: The sensor manager performs handshakes with each of the sensors, namely the accelerometer and magnetometer. Data is extracted, merged and the result is sent to the recorder object.

Recorder Object: The recorder object is tasked with controlling the recording process by listening for trigger events. The recorder streams data to and from the server and provides the BSN networking functionality.

5.5.6 Performance Optimization

While smartphones benefit from multicore processors and large amounts of Random Access Memory (RAM), a large portion of that computational power is used by the operating system and application multitasking. For example, the smartphone may run several social media application, email clients and games simultaneously. Consequently, the Skeletrix mobile application had to be designed in accordance to the computational constraints of the smartphone. Optimization was primarily focused around kinematic motion reconstruction and multithreading.

Kinematic Motion Reconstruction

Kinematic motion reconstruction is the process of applying a set of angular readings to a kinematic model and rendering the result. The kinematic motion reconstruction is achieved in the graphics pipeline by passing 3D topologies and local rotations to OpenGL ES. Figure 5-6 shows a comparison between the motion reconstruction of the Skeletrix software environment and the mobile application.

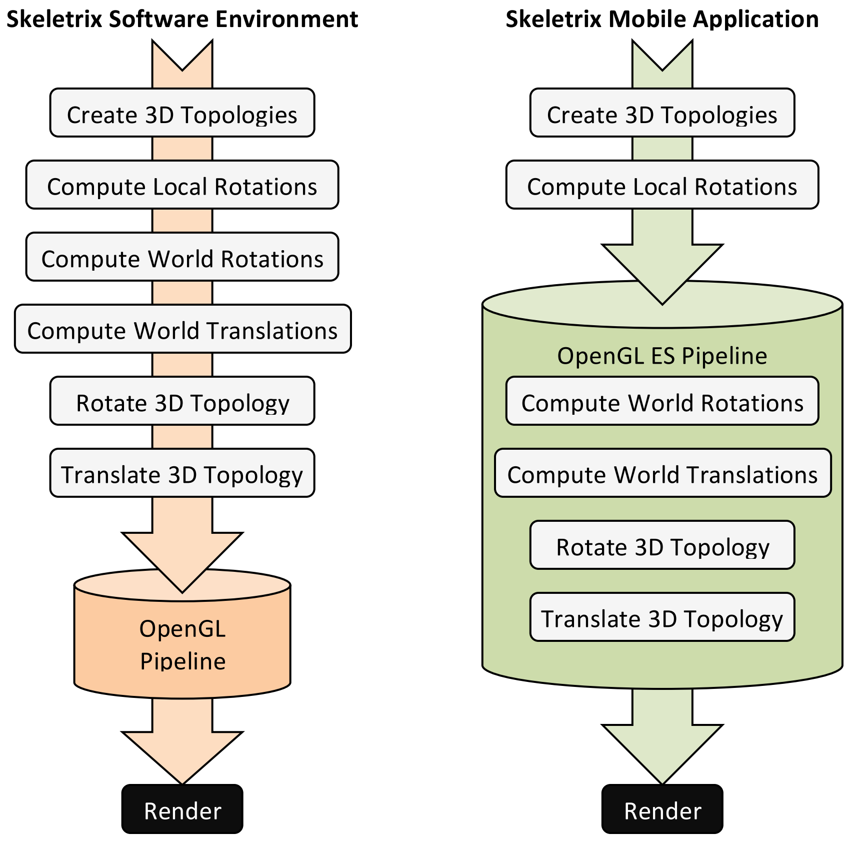


Figure 5-6: Comparison between the Skeletrix software environment and the mobile application's 3D motion reconstruction.

Multithreading

Regardless of the targeted platform, multithreading is a key aspect of any motion capture software environment. Multithreading is needed because inertial motion capture happens very quickly. Aside from extracting data from sensors, software environments need to perform other tasks such as 3D rendering, motion processing, kinematic deployment of data, etc. All those tasks tend to block the main thread, thus preventing the software from sampling the sensors at the correct time intervals. This problem is amplified in the context of software environments that use Internet connections because it is impossible to know how quickly the server will respond and the recording process cannot halt.

As shown in Figure 5-7 a thread is launched to drive the core package along with the main interface. This thread is set to tick at a low frame rate because most of the functionality is not important to the recording process. For example, multitouch gesture interaction and updating the display or loading BVH files are not urgent tasks. A second thread is launched to render motion to the OpenGL canvas at 30 f/s. The animation controller was designed to render high frequency recordings at low frame rates by skipping frames. A third thread, dedicated to extracting sensor data, is launched to run the sensor manager at frequencies specified by the BVH file in use. Generally, BVH files contain motion data at frame rates between 24 and 120 f/s.

Aside from the main three threads, asynchronous tasks are used to upload, download or poll the server. Asynchronous tasks are short-lived threads that close automatically when their task is complete.

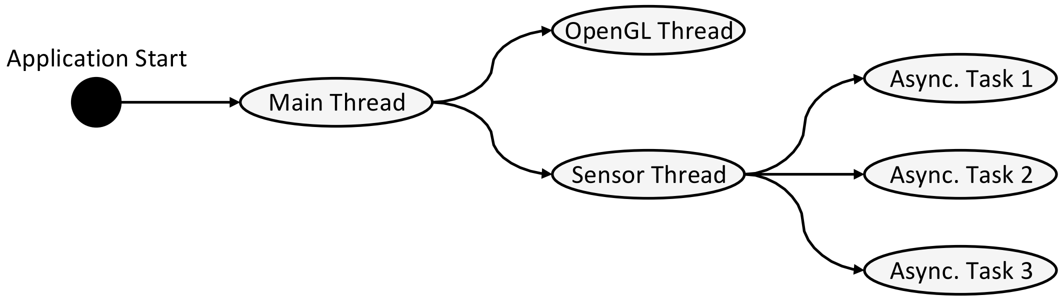


Figure 5-7: Skeletrix mobile application multithreading diagram.

5.6 Interface Overview

Condensing the functionality of the software environment to work in the context of mobile computing technologies, which have limited screen dimensions and touchscreen interfaces, proved difficult. This section discusses the functionality of the mobile application by looking at the design and development of the interface layer.

5.6.1 Use Cases

The following use case diagram, shown in Figure 5-8, was developed to meet the interface requirements of the application. Users are required to authenticate with a server by entering a unique username and password. Once authenticated, the user chooses a recording object corresponding to a BSN. Recording objects are further discussed in Chapter 6 in the context of the Motion Cloud. Once the smartphone is connected to a BSN, the user gains access to the main interface, which provides functionality for: loading kinematic models, saving kinematic models, controlling the animation, interacting with the viewport uploading and downloading data. After a kinematic model is chosen, the user can proceed to record motion.

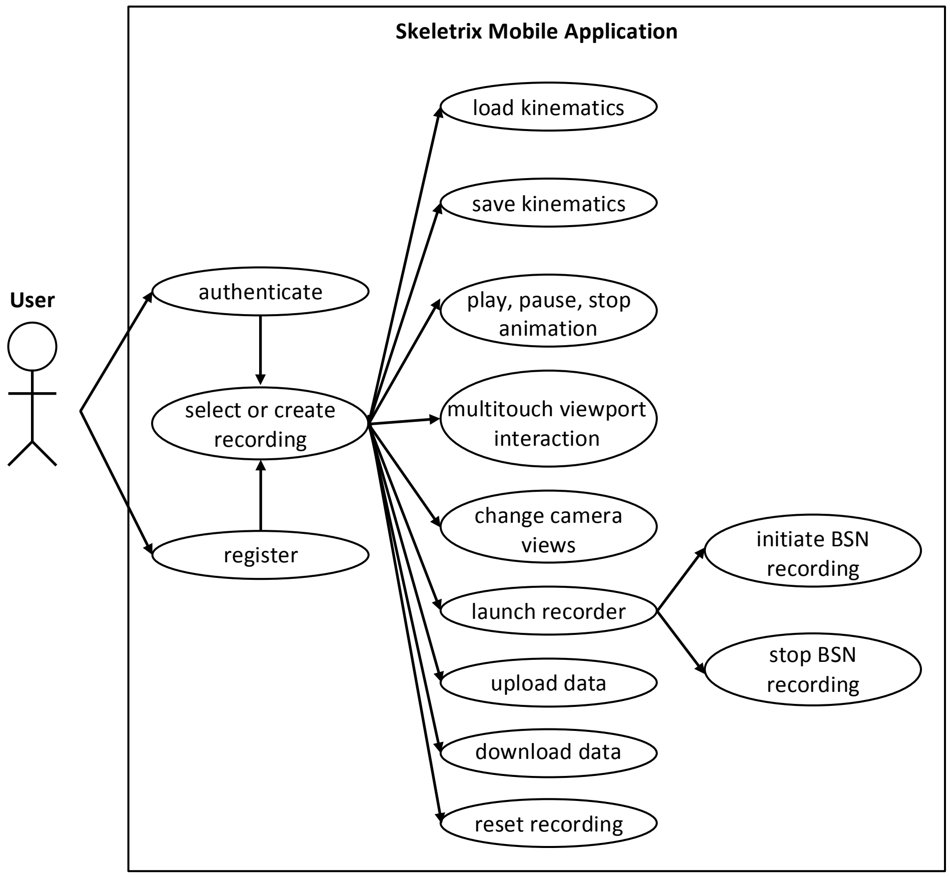


Figure 5-8: Skeletrix mobile application interface layer use case diagram.

5.6.2 Main Interfaces

The interface layer was designed efficiently by keeping the number of interface pages to a minimum. Because the mobile application is designed for the Android OS, each interface object is written in XML using the Android Interface API. As shown in Figure 5-9, the application has two main interfaces: dashboard and recorder.

The dashboard interface can be divided into three sections: toolbox, canvas and animation controls. To begin with, the toolbox is a horizontally scrollable menu located at the top of the screen allowing users to load files, save file, change camera viewports, upload or download data from the server. Each tool is represented by a clickable icon and descriptive text. Then, the canvas occupies the centre of the screen to render kinematic motion. The camera angle can be adjusted through two multitouch gestures: pinch to zoom and swipe to rotate. Lastly, the conventional animation controls can be found at the bottom of the screen.

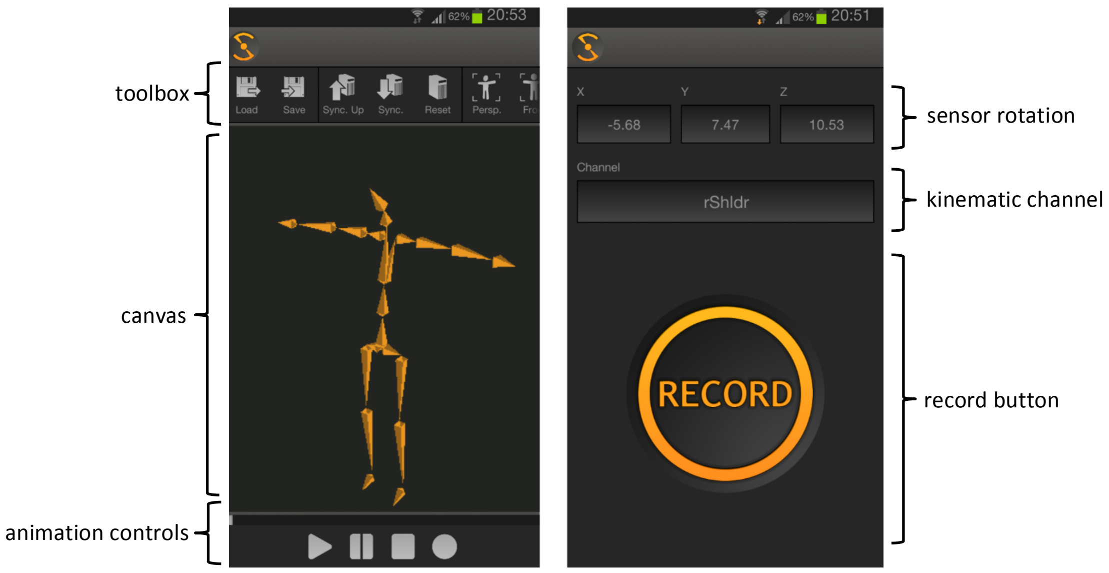


Figure 5-9: Dashboard (left) and recorder (right) diagram.

Because the dashboard is crowded with functionality, the recording process is achieved in a secondary interface. When launched, the recorder page overlays the dashboard and blocks the real-time visualisation of motion. Notably, it is difficult to see or interact with the screen when the smartphone is being worn on the body. Consequently, the recorder page's record button is large and changes colour from orange to green to give users a visual cue that motion is being captured. Additionally, the recorder page allows users to select the kinematic segment they wish to record.

5.7 Motion Processing

The first step of the mobile application workflow is motion processing. Motion processing is the act of extracting raw data from the sensors and converting it into usable motion data. In this context, data is extracted from the smartphone's accelerometer and magnetometer. The desired outcome of the motion processing stage is to produce world-space rotations as required to drive the kinematic model. The main objective of motion processing is to compute motion that contains little drift or noise. In the context of mobile computing technologies, the approach differs from that of traditional inertial motion capture system in terms of design and implementation.

As identified in Chapter 2, the main problem with motion processing is that gyroscopes have no reference axis in space and therefore suffer from drift. When computing world-space rotations, the gyroscope's accuracy decreases drastically over time.

Accelerometers and magnetometer can be combined to produce a rough approximation of rotation to give the gyroscope reference axis. However, the combined accelerometer and magnetometer data will contain a significant amount of noise whereby the signal oscillates several degrees per frame of motion. Noise can be improved with post-processing filters.

In the mobile application, motion processing is achieved in three stages: pre-processing, data fusion and post-processing using only the accelerometer and magnetometer.

5.7.1 Pre-Processing

The first step of the pre-processing stage is to extract data from the accelerometer and magnetometer, a process that is facilitated by the Android API. Optionally the gyroscope data can also be extracted. The gyroscope outputs angular speeds that require converting into world-space rotations. This process can be achieved using a similar solution to that implemented in the Motion Tracking Development System (MTDS) IMU (see Section 4.4.3). It is also important to apply a set of transformations so that the data matches the 3D engine's orthogonal configuration as specified by OpenGL ES.

5.7.2 Data Fusion

Sensor fusion is the process of merging the output of two or more sensors to produce a better result, which has little drift or noise. There are many methods to achieve this task such as the Kalman filter [171] [172] or complementary filters. Complementary filters merge the gyroscope, which has drift, with the accelerometer and magnetometer, which have noise, to produce a result that has little drift or noise.

The final version of the mobile application takes the accelerometer and magnetometer values and combines them. This process is achieved using the *getOrientation* function call of the Android API. This approach is desirable for situations where older smartphones, that do not enclose gyroscopes, are used to record motion. However, using the accelerometer and magnetometer requires post-processing.

5.7.3 Post-Processing

Post-processing of inertial motion data is commonly referred to as data cleaning. The mobile application cleans the motion data in two stages: smoothing and error checking. A smoothing filter, which applies a Gaussian mask to the motion data, is used to remove unwanted noise. Notably, the smoothing filter is optional because it can reduce accuracy in favour of creating better-looking animation. Errors, which are miscalculated angular readings, can occur due to sensor or software problems. It is therefore important to check every angular reading against its neighbours and, if required, replace the angular reading with an interpolated substitute.

After the motion data is cleaned, the mobile application applies smoothing to all the angular readings. Smoothing is an optional process whereby 2D Gaussian masks are applied to every frame of motion. Smoothing is optional because it compromises accuracy to produce a result that looks good. However, smoothing is desirable because the application fuses only the accelerometer and magnetometer data, thus producing a result that contains large amounts of noise. Raw data may be preferred for scientific studies (because it is more accurate) while fluent motion may be desired for animation (because it looks better). Although it is not implemented in the interface, the mobile application can be configured to enable or disable the amount of smoothing depending on the task.

5.8 Event Triggers

Although an animation suit consists of several hardware devices networked together, the hardware behaves as a singular device. Consequently, an important element of a BSN is the control mechanism that allows multiple devices to be controlled by one centralised node that is often referred to as a multiplexer. The smartphone-driven BSN replaces the concept of a multiplexer for a server and the control mechanism needs to be implemented at the server level. In terms of usability, interacting with several body-worn smartphones simultaneously is difficult. The smartphones must be remote controlled using event triggers. The term event trigger describes a specific web service that sends events to multiple smartphones concurrently. Because the design of the BSN is omnidirectional, an event trigger may be accessed from the any networked smartphone's interface.

This example implementation of event triggers uses a passive server that cannot actively contact a smartphone. Each BSN's online data is accompanied by a trigger value, which is represented by an integer. The trigger value can be modified by any networked smartphone through a web service. At the same time, every other smartphone listens to that integer for changes. If a change is discovered, the mobile application reacts accordingly. Figure 5-10 illustrates how multiple smartphones can be controlled from one interface. The white smartphone is tapped to begin recording, and after the black smartphones replicate that functionality. The status change is made visible as the interface changes colour from orange (inactive) to green (active).

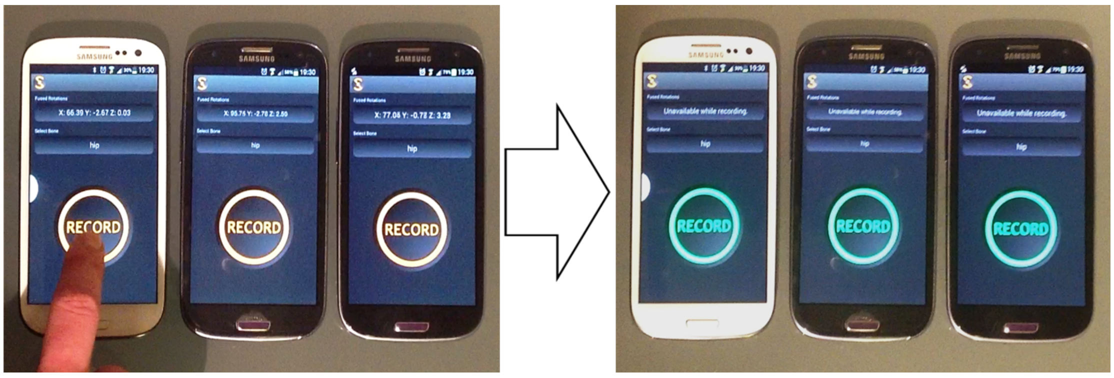


Figure 5-10: Remote-controlling smartphones using event triggers.

Source: Pascu et al. [15] [16]

5.9 Data Communication Protocols

An important aspect of the smartphone-driven BSN is data intercommunication between smartphones. While inertial motion capture suits use short distance connections such as Bluetooth, a smartphone connection can vary between three states: fast connectivity (i.e. Wi-Fi or 4G), slow connectivity (i.e. 3G) and no connectivity. Additionally, the signal strength may vary (e.g. in some circumstances making a 4G connection slower than a 3G connection). In many situations the connection will alternate between different modes.

A flexible data communication protocol is needed to ensure that the smartphone can stream data to the Motion Cloud. The problem is not the quantity of data but the frequency at which a connection to the Motion Cloud is established. The proposed data streaming protocol allows the smartphone to send motion data to the server in variable-sized buffers. The size of the buffers is governed by the connection speed. The mobile application buffers data and starts asynchronous server calls. Data is uploaded by each asynchronous call in the form of JavaScript Object Notation (JSON) packets.

Fast Connection

In situations where the connection is fast, asynchronous calls are made as frequently as possible for real-time data streaming. The highest frequencies will likely be achieved over Wi-Fi.

Slow Connection

In situation where the connection is slow, the mobile application begins streaming data in larger chunks less frequently. Over 3G, the smartphone will upload data to the server approximately twice per second depending on signal strength.

No Connection

In situation where the connection drops, the mobile application begins recording motion into local storage. When a connection is eventually regained, all the buffered data is uploaded. This approach ensures that no motion data is lost.

5.10 Synchronization

Synchronization primarily occurs once all the smartphones have uploaded their data to the Motion Cloud repository as channel objects. Synchronization is required as each smartphone begins and finishes recording at slightly different times depending on connection latencies. The beginning and end of each channel requires truncation to correct data misalignments. Figure 5-11 illustrates the simulated recording process of five smartphones for a short period of three seconds at the frequency of 10f/s. The darker rectangles are used to illustrate alignment.

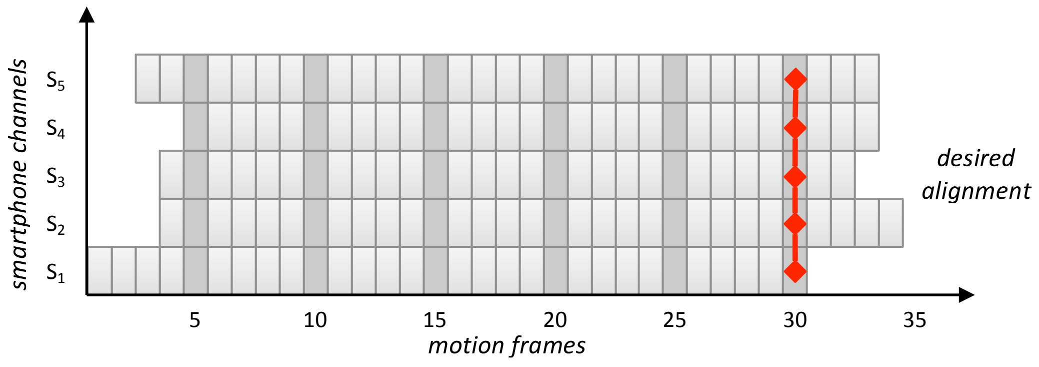


Figure 5-11: Smartphones starting and stopping the recording process through event triggers at varying times depending on connection latencies.

Figure 5-12 illustrates how the motion data would be interpreted by smartphones without synchronization. The channel contents are pushed to the left because the BSN has no means of knowing when each smartphone has started or finished recording. The result is a misaligned set of data that can be aligned through synchronization.

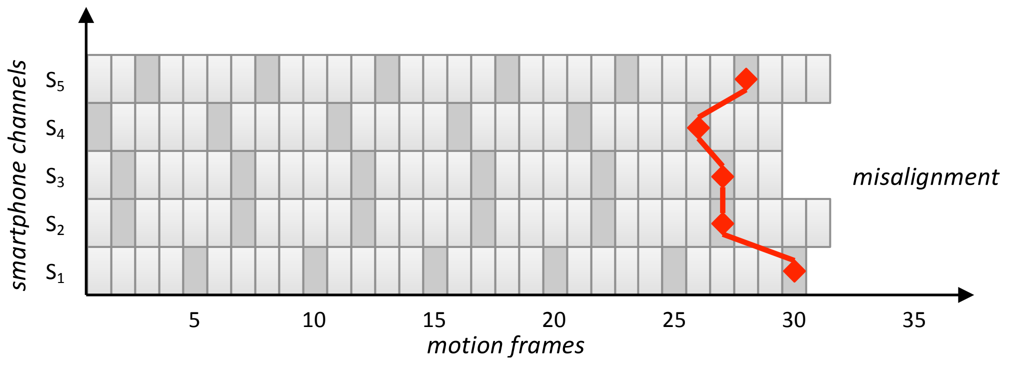


Figure 5-12: Smartphones interpreting unsynchronized data.

The proposed methodology for synchronization consists of three solutions for: pre-synchronization, post-synchronization and capping.

5.10.1 Pre-Synchronization

Pre-synchronization is a simple process that takes place at the beginning of a recording session. Every channel in a recording is emptied when a smartphone is tapped to begin recording. As shown in Figure 5-11, smartphone S_2 begins recording after S_1. At the point where S_2 has begun recording, S_1's channel already contains several frames of motion that will cause a misalignment in the data. Consequently, S_2 empties S_1's channel to synchronize the data. As shown in Figure 5-13, this approach improves the alignment of the data although the result is not perfect.

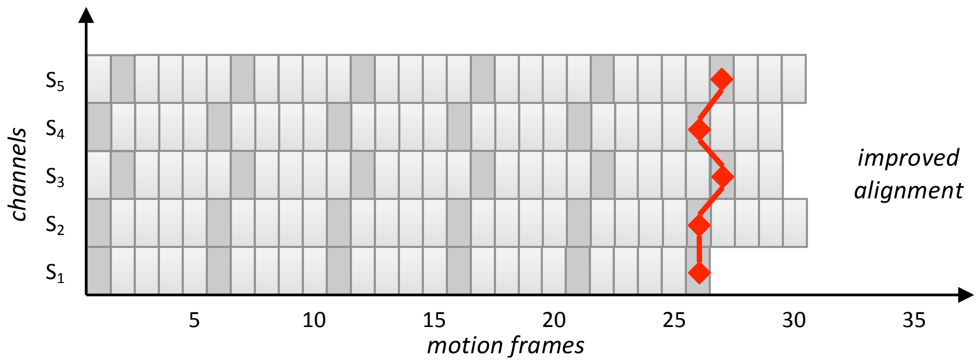


Figure 5-13: Alignment of channel data using the pre-synchronization methodology.

5.10.2 Post-Synchronization

While the above methodology shows a significant improvement, the improvement is only noticeable if the connection is slow. Post-synchronization takes place after a recording session has finished. To synchronize the channels further, the smartphones takes advantage of the smartphone's clock based on the assumption that two Android devices in the same time zone have the same or at least a very similar clock reading. Depending on how accurate the clock is, post-synchronization could improve data alignment. At the beginning of a recording session, each smartphone uploads a timestamp corresponding to the exact time it began recording. Table 5-1 illustrates the simulated timestamps of the five smartphones in the above examples.

Table 5-1: Timestamps corresponding to the beginning of the recording session.

Smartphone Channel	Timestamp (ms)	Difference (ms)	Truncated frames
S_1	58752245	0	0
S_2	58752571	326	3
S_3	58752502	257	3
S_4	58752612	367	4
S_5	58752399	154	2

The latest timestamp is used as a reference point to truncate any unwanted motion frames. The smallest timestamp is subtracted from each channel's timestamp. The resulting difference is divided by number of milliseconds between two motion frames depending on the recording's frame rate, to determine the number of motion frames that require truncation. As shown in Figure 5-14, the server iterates through the channels and removes the concluded amount of motion frames to synchronize the data. The desired result is a synchronized recording whereby the data is aligned correctly. Truncating motion frames implies removing data that could useful. An alternative and possibly better methodology would be to insert blank frames.

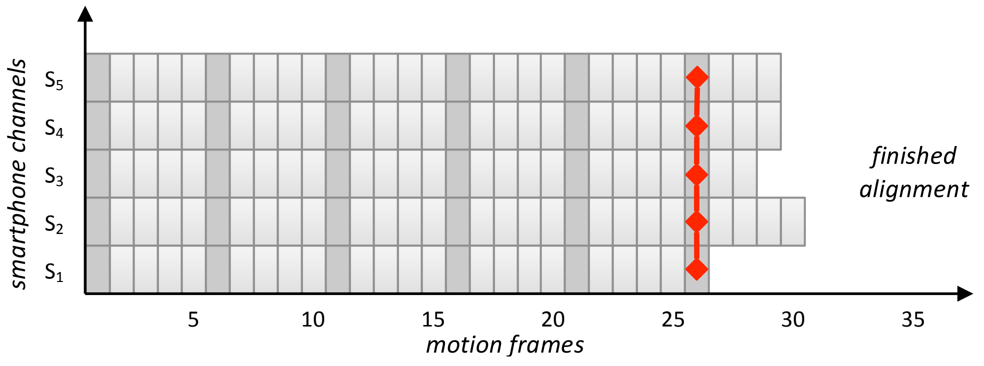


Figure 5-14: Post-synchronization alignment of channel data.

5.10.3 Capping

As illustrated in Figure 5-15, the final stage of synchronization is to truncate the end of motion channels such that all channels are the same length. This step is required primarily for the mobile application's parser to avoid potential errors.

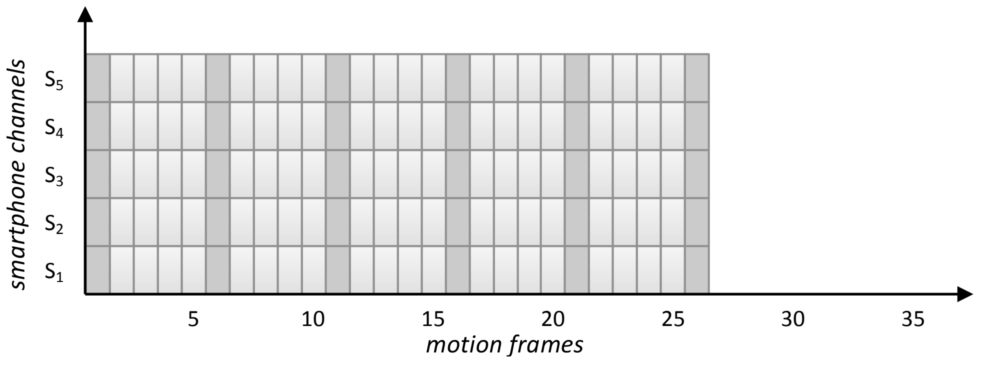


Figure 5-15: Synchronized result.

5.11 Versioning

The development of the mobile application was achieved in incremental stages whereby each stage is focused on demonstrating part of the functionality. As shown in Table 5-2, this section discusses the five release versions, each having multiple subversions.

The first version was focused on achieving core functionality that proved the concept to be viable. Data was extracted from the gyroscope, applied to a kinematic model and visualised in OpenGL ES and outputted as a BVH file. The second version was focused on evaluating the accelerometer and magnetometer while improving the overall interface with multitouch interaction and adding animation playback functionality. The third version was focused on adding networking functionality through web technologies whereby users can upload and download data from the server. The fourth version introduces data streaming, a process that required a redevelopment of the multithreading architecture. The final version brings all the previous developments into one large package.

Table 5-2: Skeletrix mobile application versioning.

Functionality	1.0	2.0	3.0	4.0	5.0
Core Functionality					
Record Data (Gyro)	**YES**	**YES**	**YES**	**YES**	**YES**
Record Data (Acc/Mag)	N/A	**YES**	**YES**	**YES**	**YES**
Motion Processing	N/A	N/A	N/A	**YES**	**YES**
Kinematics	**YES**	**YES**	**YES**	**YES**	**YES**
Motion Files Parsing	**YES**	**YES**	**YES**	**YES**	**YES**
Local Storage of Motion	N/A	**YES**	**YES**	**YES**	**YES**
Interface Functionality					
OpenGL Rendering	**YES**	**YES**	**YES**	**YES**	**YES**
Multitouch Interaction	N/A	**YES**	**YES**	**YES**	**YES**
Camera Selection Tools	N/A	N/A	**YES**	**YES**	**YES**
Animation Playback	**YES**	**YES**	**YES**	**YES**	**YES**
Web Functionality					
Web Storage of Motion	N/A	N/A	**YES**	**YES**	**YES**
User Authentication	N/A	N/A	**YES**	**YES**	**YES**
User Registration	N/A	N/A	N/A	**YES**	**YES**
Create Recording Objects	N/A	N/A	N/A	N/A	**YES**
Upload Motion	N/A	N/A	**YES**	**YES**	**YES**
Download Motion	N/A	N/A	**YES**	**YES**	**YES**
Stream Motion	N/A	N/A	N/A	**YES**	**YES**
BSN Functionality					
Event Triggers	N/A	N/A	**YES**	**YES**	**YES**
Data Synchronization	N/A	N/A	N/A	**YES**	**YES**

5.12 Capturing Motion

The mobile application was evaluated in three stages whereby each stage demonstrates several aspects of the mobile application's functionality. While the first stage involves simulating a hand wave gesture on a flat surface, the second stage focuses on capturing a body's motion in the traditional way by strapping three smartphones to the body. The third stage demonstrates the mobile application's ability to sense continuously through an example of activity tracking. These results have previously been published in Pascu et al. [15] [16] [19].

5.12.1 Simulating a Hand Wave

As shown in Figure 5-16, the first and most simple test of the mobile application was performed on a flat surface using two networked Samsung Galaxy S3 smartphones. By sliding the two smartphones in an arch fashion, the kinematic model simulates a simple hand wave gesture.

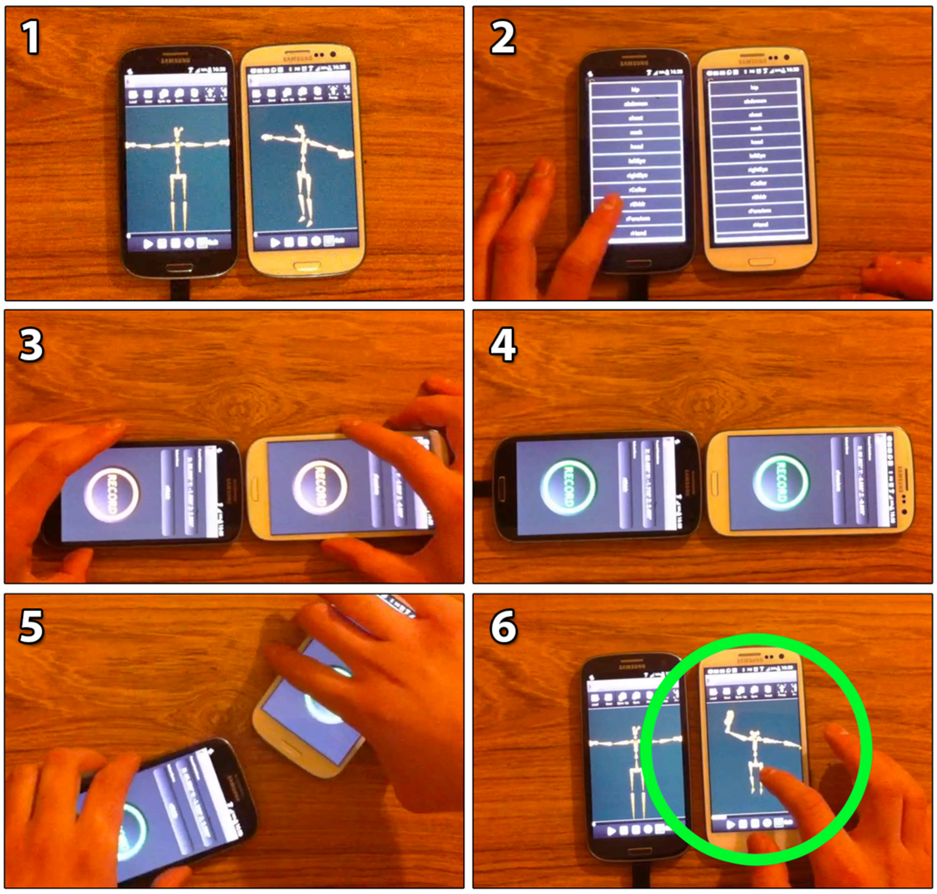


Figure 5-16: Simulating and capturing a hand wave gesture using two Samsung Galaxy S3s.

Source: Pascu et al. [15] [16]

This study demonstrates the mobile application's user journey through six user-performed steps:

1. *Load Kinematic Models:* The mobile application is launched on both smartphones and identical BVH files are loaded to generate the same kinematic model on both devices.

2. *Select Kinematic Joints:* The recorder is opened and a kinematic segment is chosen. The black smartphone selects the *rShldr* joint (corresponding to the arm) and the white smartphone selects the *rArm* joint (corresponding to the forearm).

3. *Perform Calibration Gesture:* The smartphones are placed horizontally for calibration whereby the difference between the kinematic model and the device's rotation is compensated. This step works in a similar way to T-Pose calibration in motion capture suits.

4. *Begin Recording:* One of the smartphones is tapped to begin recording and both interfaces turn green, thus demonstrating the concept of event triggers.

5. *Perform Animation:* The smartphones are slid on the flat surface in an arch fashion to simulate a basic hand wave gesture consisting of two consecutive waves. Once the animation is completed, the smartphones are returned to their horizontal configuration.

6. *Visualise Results:* Immediately after the smartphones are tapped to stop recording, the data is uploaded to the server and downloaded on each device. By pressing the play button, the result (highlighted in green) can be visualised on both smartphones.

5.12.2 Recording a Hand Wave Gesture

Having proved the concept that smartphones can be used to record articulated motion; the next step involved using smartphones to record real human motion. Three Samsung Galaxy S3 devices were strapped to the motion performer's right arm to form a motion capture sleeve. More specifically, a smartphone was strapped to the arm, forearm and hand using elastic bands. This particular device is ill suited for this type of application due to its large screen dimensions. However, because the device has a large surface area and is relatively light, it slides very little in relation to the body.

During the experiment, the same recording object and kinematic rig were loaded on each of the three devices. The recorder was launched and the devices became networked. The recording frequency was set to 30f/s and the right arm was extended laterally away from the body to perform the T-pose. The record button was tapped and all devices started recording.

As with the previous experiment, the performed gesture is a wave gesture of three consecutive swings of the right arm. This gesture was chosen for the experiment because it provides a clear image of the data. If a sensor is not synchronized or drifting, the error is immediately visible in the 3D reconstruction. Figure 5-17 shows a comparison between the real-life movement and its corresponding virtual reconstruction.

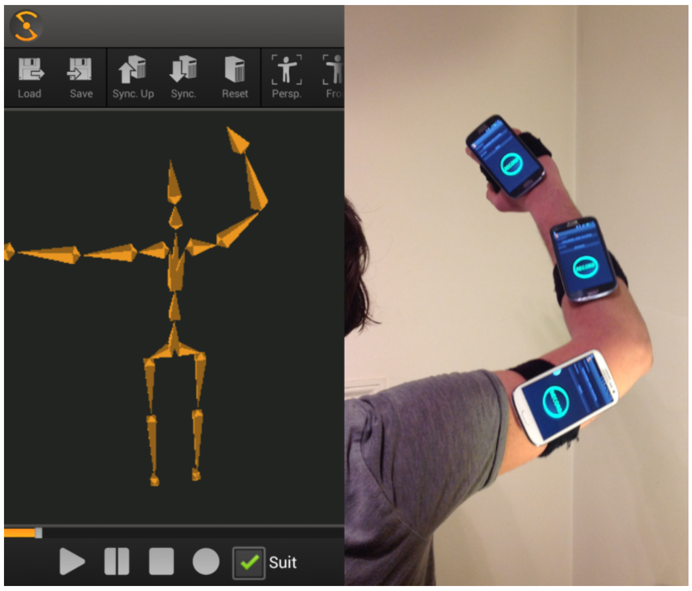


Figure 5-17: Capturing an arm's motion using three body-worn Samsung Galaxy S3 smartphones. **Source: Pascu et al [19]**

Figure 5-18 shows the rotational data recorded during the gesture. The graph shows a repeating pattern between each kinematic channel and between consecutive waves. Throughout the gesture, the arm rotates approximately 20 degrees in each of the three axes. The forearm and hand move in tandem, rotating in excess of 100 degrees. From the graphs we can determine that the arm does the least movement while the forearm and hand have produce nearly indistinguishable data sets.

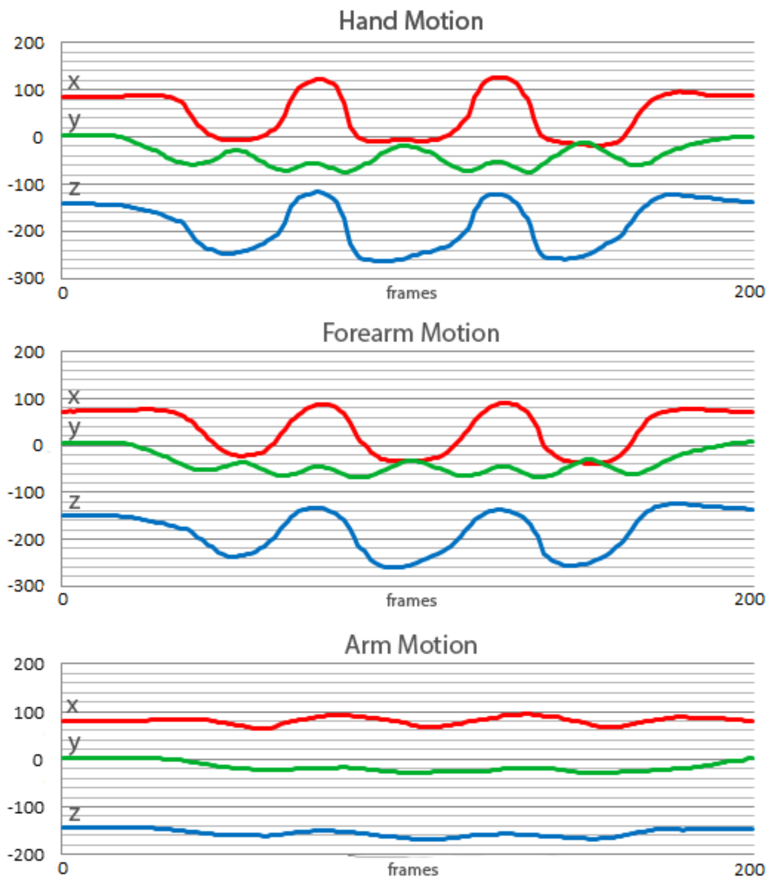


Figure 5-18: Rotational data of three consecutive hand wave gestures.

Source: Pascu et al. [19]

5.12.3 Recording Active and Sedentary Behaviour

The previous examples have demonstrated BSNs worn by one person. In this activity-tracking example, the BSN receives data from two separate individuals performing everyday activities over a time period of two hours. This experiment was designed to put stress on the mobile application's data streaming protocol to see if continuous sensing is possible, if the mobile application crashes and if data is lost.

To put the data into context, the smartphones are used to track the physical activity of a sedentary individual in the work environment and a more active individual going walking through a town's centre. Both recordings take place simultaneously. While the first individual records motion through Wi-Fi, the second individual alternates between Wi-Fi, 3G and no connection. The BSN is used to aggregate and compare the data. The combined rotation per frame of motion is summed to conclude a coefficient of activity based on the assumption that if a person is more active, the smartphone will rotate more. Figure 5-19 show a comparison between the two sets of data where sedentary (green) behaviour is superimposed on top of active (grey) behaviour.

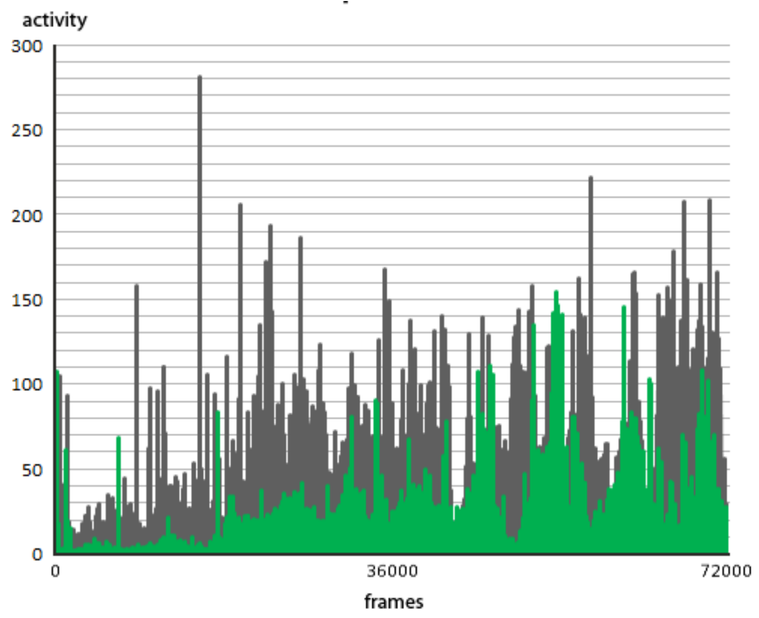


Figure 5-19: Graphical comparison between sedentary (green) behaviour and active (grey) behaviour. **Source: Pascu et al. [19]**

While there are better methodologies for capturing activity data, these results demonstrate that the mobile application is abled to track motion continuously during everyday use. The task of incorporating a complex activity tracking system extends beyond the scope of this thesis. While activity tracking is generally achieved using accelerometer data, this example demonstrates that rotations can be used for this purpose. The mobile application could be calibrated against a commercial pedometer (e.g. FitBit [173], Jawbone UP [174], etc.) to compute the number of steps taken, calories burnt and distance walked.

5.13 Conclusions

Smartphones have a significant impact the field of inertial motion capture and BSNs because, like IMUs, smartphones contain gyroscopes, accelerometers and magnetometers along with telecommunication technologies. This chapter draws a parallel between the smartphones and IMUs to investigate the sensing capabilities of mobile computing technologies. While the concept of smartphone sensor networks is not new to smartphones, the act of creating an omnidirectional smartphone-driven BSN for human motion tracking is.

Within the scope of this thesis, the proposed mobile application is developed from the Skeletrix software environment presented in the Chapter 3. Unlike the Skeletrix software environment, where motion processing is made transparent to the users, the mobile application aims to automate the process to achieve a minimalistic user journey. This chapter investigates the implications of simplifying, porting and heavily modifying the Skeletrix software environment on the Android platform.

The task of constructing an omnidirectional smartphone-driven BSN that uses web technologies was achieved through several key developments, all of which have been discussed throughout this chapter: motion processing, real-time data communication protocols, mechanisms for remote-controlling the functionality of network nodes, data synchronization, etc. Learning what smartphones can do can be beneficial to developing better inertial motion capture systems. Rather than defining the BSN configuration at the hardware level through a physical multiplexer (as demonstrated by MTDS), the BSN can now be conceptualised and created on a web server through software. For example, the number of nodes belonging to the BSN is no longer limited by the multiplexer. Smartphones can be added or removed with ease, thus allowing users to customize the BSN for their individual experiments.

5.13.1 Application Areas

This section discussed four potential usability scenarios that illustrate the versatility of the mobile application. The four proposed usability scenarios are: medical science, activity tracking, emergency responses, road and traffic condition monitoring.

Scenario 1: Medical Science

As published in Pascu et al. [15] [16], a very important application area for inertial motion capture is medical science. Some medical disorders have a measurable effect on the motor functions of the body. Smartphones can be used to track the motion of one or several points on the body. In the case of idiopathic scoliosis [51], a torso-mounted device can determine the body's axis of balance and identify musculoskeletal asymmetries. In the case of Parkinson's disease [48] [79], a limb-mounted device can measure symptoms such as shaking, body rigidity and slowness of movement. A body-worn smartphone has the potential to measure treatment responses and while providing useful data to medical and healthcare professionals.

Scenario 2: Activity Tracking

In the context of health and fitness, the proposed mobile application can be used to compute a rough estimation of physical activity. The smartphone presents the necessary hardware to replicate the functionality of commercial pedometers. While this chapter has demonstrated basic activity tracking from kinematic rotations, there are more accurate solutions for tracking physical activity such as Samsung S-Health [175]. The application has the potential to be extended with algorithms for converting rotational data into activity parameters. For example, [176] shows a much more thorough approach for turning the smartphone into a pedometer using motion data.

Scenario 3: Emergency Responses

The mobile application allows the smartphone to stream motion data continuously into an online repository. That data can be processed to identify anomalies that are indicative of emergency situations. For example, inertial sensors could be used to detect if an elderly person has collapsed [177] or if someone has experienced an accident and notify authorities.

Scenario 4: Road and Traffic Condition Monitoring

The mobile application could be used outside the context of character motion tracking to record the motion of vehicles. In vehicles, the smartphone is commonly used either as a storage device for music or for GPS turn-by-turn navigation. Sensing the motion of the vehicle could be beneficial in monitoring road and traffic conditions [178]. For the mobile application to be used in this context, additional functionality is required to interpret the motion data and potentially merge it with geographical location data.

CHAPTER SIX

6 Motion Cloud: A Repository and Gateway for Inertial Motion Data

6.1 Introduction

The focus of this chapter is the Motion Cloud as published in Pascu et al. [19], an online repository and gateway for all motion capture mediums: animation suits, smartphones, inertial measurement units and even activity tracking accelerometer-based devices. The Motion Cloud elaborates the concepts of storing, organising and accessing motion data through an innovative set of web services and database models. The proposed architecture is highly extendable and optimized to handle large amounts of data. The research work presented in this chapter can be divided into three main categories: repository, gateway and web portal. While the repository and gateway are integral to the Motion Cloud technology, the web portal is an auxiliary component providing an interface layer. The interface layer is used to test the repository and the gateway and allows users visualise and manipulate the motion data.

With the research work presented in this chapter, the Skeletrix software environment and mobile application are given functionality for communicating (streaming or uploading) inertial motion capture data into online storage. This concept was first introduced in the previous chapter in the context of online BSNs where the multiplexer is substituted for an online server.

Unlike other motion libraries, the Motion Cloud is unique because it provides a more detailed methodology for interacting with the motion data. Each rotational value produced by an inertial motion capture system is stored as a single entry in the database. Therefore, the data can be accessed quickly through the web portal or through web services by software applications. This gives the Motion Cloud architecture a great potential to be extended to contain motion analysis and cleanup tools, thus allowing users to inspect or edit the data online. This is not currently possible with existing online motion databases that contain the motion data uploaded as files, which must be downloaded to be processed.

6.2 Framework Relevance

The Motion Cloud can be seen as a bridging architecture between the two previously introduced software components of the Skeletrix framework, namely the software environment and the mobile application. The act of connecting the two framework components has positive ramifications. For example, all inertial technologies (e.g. smartphones, BSNs, inertial measurement units (IMU), development boards, pedometers, etc.) are treated equally allowing for performance evaluations and benchmarking. The Motion Cloud creates a methodology for transferring motion data between the software environment and the mobile application presented in this thesis.

The motion capture workflow, which was previously limited to extracting data from hardware using a computer, is expanded to include web technologies. As shown in Figure 6-1, web technologies are the top layer of the proposed framework. To make the Motion Cloud concept relevant beyond the scope of this framework, the gateway is developed as an Application-Programming Interface (API) allowing for third-party software integration (e.g. downstream applications that can download motion data from the repository).

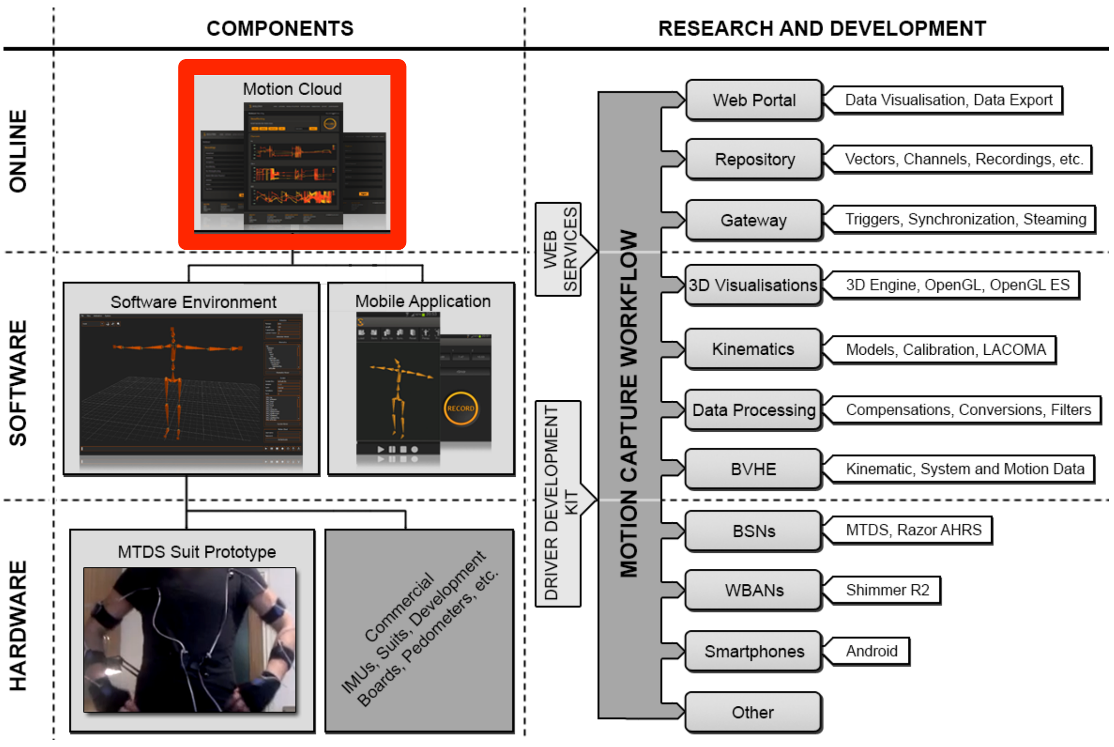


Figure 6-1: The Motion Cloud adds web technologies to the Skeletrix framework.

Source: Pascu et al. [15] [16]

6.3 Requirements Specification

At its core, the design process of the Motion Cloud is focused on three principal areas: repository, gateway and web portal. This section summarizes both functional and non-functional requirements to create a basic development specification.

6.3.1 Repository

Object-Oriented Data Model: Object-orientation provides an efficient approach to organising data so that it is easily accessible. Therefore, the data model should be object-oriented whereby a BSN's output can be represented as a set of data model objects. Each data model object must have a logical purpose and exist both independently and as a component of a hierarchy. In the context of inertial motion capture, the hierarchy will mimic the skeletal definition of the system in use. For example, an upper body BSN, like the Motion Tracking Development System (MTDS), will produce seven channels objects, each containing multiple vector objects, of motion data in the Motion Cloud.

Data Storage: Motion capture data streamed from a BSN, even over short periods of time, implies a large number of database entries. Every angular reading must be placed as an object in the database so that it may be identified and retrieved later. Additionally, the data model must be optimized to keep the database size to a minimum. As previously discussed in Section 2.7.1, stored motion data must be in a format that is both intuitive and easy to extract from hardware. While quaternions and rotation matrices provide performance, only the Euler rotational model can be interpreted (through 2D graphs) without using 3D visualisations.

Object Dependencies: All data stored in the repository hierarchy should conform to a list of dependencies. If a user deletes a channel, all hierarchy dependent vector model objects must also be removed. If a user uploads a recording, the repository must automatically generate and organize new data model objects.

User Accounts: The repository should differentiate between its users to protect the stored data. To use the Motion Cloud, users are required to first register an account by providing personal information such as profile information and authentication details. While security and privacy are important, this requirement is not mandatory for a prototype-level development. However, each motion recording must be bound to one user account that is protected by a password.

6.3.2 Gateway

Data Streaming: Because inertial motion capture systems do not store data internally, the gateway should allow systems to stream data to the repository. In theory, data can be streamed as it is obtained from the hardware, one angular reading at a time. In reality, data is buffered and streamed as packages, which contain several angular readings each. Data streaming involves asynchronous server calls made by client applications (e.g. software environments, drivers, etc.) to upload the buffered data as packets.

Application-Programming Interface: The Motion Cloud is designed and implemented as an online platform for inertial motion capture data that can be integrated with third-party software applications. An API consisting of generic web services and documentation is required to allow software developers to interface with the gateway. It is desirable for the web services to be versatile and to facilitate most usability scenarios.

6.3.3 Web Portal

Data Visualization: Graphs provide an instant overview of the motion whereby users can determine if an IMU is producing valid data and if the data contains drift and noise. Graphical spikes can provide a preliminary understanding of the recorded motion. For example, frequent highpoints and depressions would indicate active behaviour while flatter curves would suggest sedentary behaviour. Therefore, a requirement for the web portal is to generate graphs from motion data.

Data Retrieval: The web portal should provide the functionality for exporting data in tabular or motion capture file formats. For example, an animation suit's data that is streamed into the repository could be downloaded directly by the user in a tabular format.

Data Manipulation: Data manipulation implies accessing, editing or deleting objects stored in the database. Through the web portal, all five layers of objects (i.e. profiles, groups, recordings, channels and vectors) must be made accessible and editable through the interface so that users have full control of their motion data.

6.4 Repository Design and Implementation

The repository is designed to accommodate any configuration of sensors, whether it's a single IMU streaming one channel of data or a full-body animation suit streaming eighteen channels. Therefore, the data model is both object-oriented and extensible to grow and shrink depending on the hardware output. The data model was designed and implemented using a set of abstraction layers whereby the user is the most abstract layer and vector objects are the least. As shown in Figure 6-2, the repository contains primarily five objects that form a hierarchy: vector, channel, recording, groups and profiles.

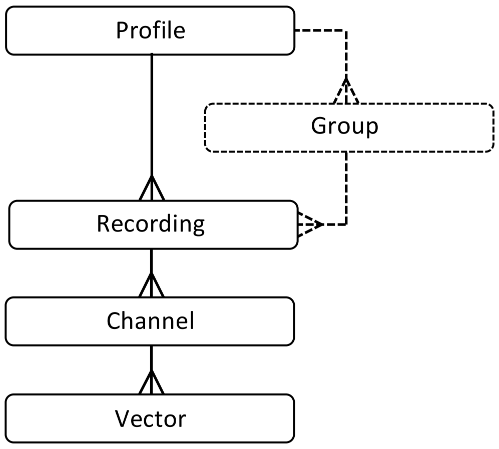


Figure 6-2: Repository data model object hierarchy.

6.4.1 Vector Objects

Vectors are small table objects that store three floating-point numeric values corresponding to a sensor's yaw, pitch and roll. Although the focal topic of this thesis is inertial motion capture, vectors could be used to cover other types of data such as positional vectors, gravitational forces, magnetic field readings, etc. For example, there is potential for other motion capture technologies, such as Microsoft Kinect, to store data in the same data model. This concept was evaluated by integrating the FitBit pedometer and storing its activity data, as vector objects, in the repository. Towards the end of this chapter, the concept of integration is discussed by developing a MotionCloudViewer for the software environment and a standalone driver for the Razor IMU presented in Chapter 4.

6.4.2 Channel Objects

Channel objects enclose multiple vectors that represent a stream of recorded data where each vector is a single sensor reading from one BSN node. A single smartphone will only use one channel. A full-body animation suit will contain approximately twenty channels if the hardware's data is uploaded directly. If the data is first passed through a kinematic model, the number of channels will double. This is because kinematic models contain dummy skeletal segments, which rotate and translate purely though kinematic constraints. The Motion Cloud would interpret those dummy skeletal segments as empty channels. For example, the motion performer may wear only one IMU on their back while the kinematics skeleton has an articulated spine consisting of several dummy skeletal segments.

6.4.3 Recording Objects

A recording, as the title would imply, contains all the data produced by a BSN from when it was activated to when it was switched off. Recording objects contain one or more channels whereby each channel corresponds to one BSN node. Additionally, recording objects contain a useful text description and title that can be used to label the performed motion. For example, during a behavioural experiment, the system operator can write the motion performer's name in the recording title and outline the recording circumstances in the text description.

6.4.4 Group Objects

Groups are simple data structures enclosing one or more recording objects. Unlike the other levels of the hierarchy, groups are not integral to the storage mechanism. Once authenticated, the user can see all the recordings bound to his account or alternatively, only the ones belonging to a group. The group object can also be given a text description to summarise its contents. The concept of organising recordings in groups is open to interpretation. A group may contain all the recordings completed using a particular hardware configuration (e.g. all the motion data recorded using the MTDS system). For researchers, a group may include all the recordings relating to a particular experiment. This functionality was designed for research projects like the ones presented in Section 1.6, namely the Motion in Place Platform project (motion tracking of archaeologists) [25] [26] [27] and the forensic psychology experiment (motion tracking of burglars) [30] [31] [32]. These research projects involved capturing several actions performed by actors, some of which required multiple takes. The group object simplifies the process storing, organising and categorising data in these scenarios.

6.4.5 Profile Objects

The highest and most abstract level of the data model hierarchy is the profile object. Anyone using the Motion Cloud must first register a user account and authenticate to be granted access to the data. The repository's user object stores authentication details, first name, last name and contact details.

6.4.6 Data Model Specification

The following database schema, shown in Figure 6-3, was used as a development specification for the data model. It summarises the objects previously discussed and showcases their fields.

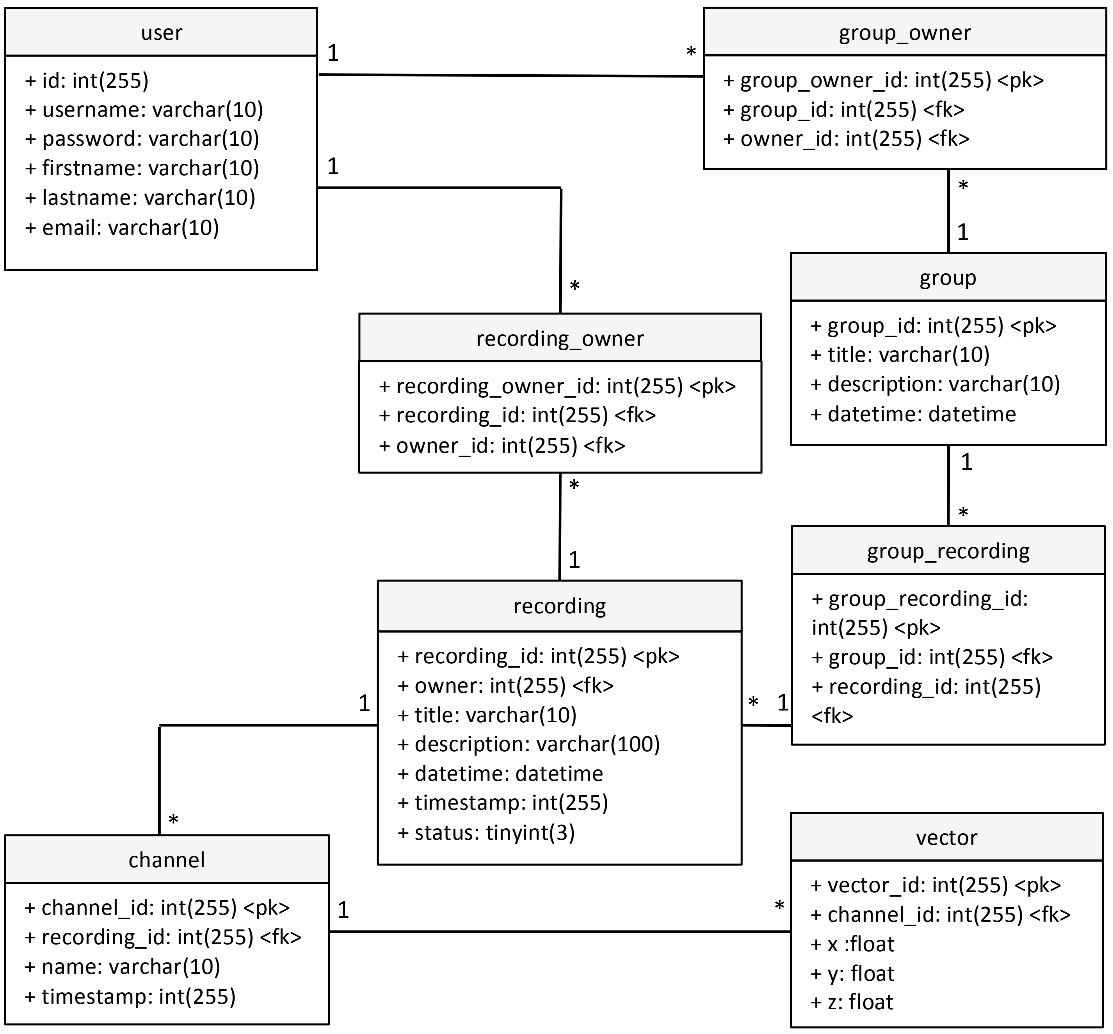

user
+ id: int(255)
+ username: varchar(10)
+ password: varchar(10)
+ firstname: varchar(10)
+ lastname: varchar(10)
+ email: varchar(10)
group_owner
+ group_owner_id: int(255) <pk>
+ group_id: int(255) <fk>
+ owner_id: int(255) <fk>
group
+ group_id: int(255) <pk>
+ title: varchar(10)
+ description: varchar(10)
+ datetime: datetime
recording_owner
+ recording_owner_id: int(255) <pk>
+ recording_id: int(255) <fk>
+ owner_id: int(255) <fk>
recording
+ recording_id: int(255) <pk>
+ owner: int(255) <fk>
+ title: varchar(10)
+ description: varchar(100)
+ datetime: datetime
+ timestamp: int(255)
+ status: tinyint(3)
group_recording
+ group_recording_id: int(255) <pk>
+ group_id: int(255) <fk>
+ recording_id: int(255) <fk>
channel
+ channel_id: int(255) <pk>
+ recording_id: int(255) <fk>
+ name: varchar(10)
+ timestamp: int(255)
vector
+ vector_id: int(255) <pk>
+ channel_id: int(255) <fk>
+ x :float
+ y: float
+ z: float
1
*
1
*
*
1
1
*
*
1
1
*
* 1
1
*

Figure 6-3: Database schema of the repository data model.

6.5 Gateway Design and Implementation

Having defined a repository for storing BSN data, the next component of the Motion Cloud is the gateway. As shown in Figure 6-4, the gateway is an access layer for connecting the repository with software-environments, mobile applications, drivers, etc. The purpose of the gateway is to allow software developers to integrate third-party applications with the Motion Cloud. The gateway features a flexible API consisting of a new set of web services allowing developers to create, retrieve, update and delete repository objects. Using the API, users develop motion capture software applications that communicate data to and from the Motion Cloud and benefit from its features (e.g. the web portal).

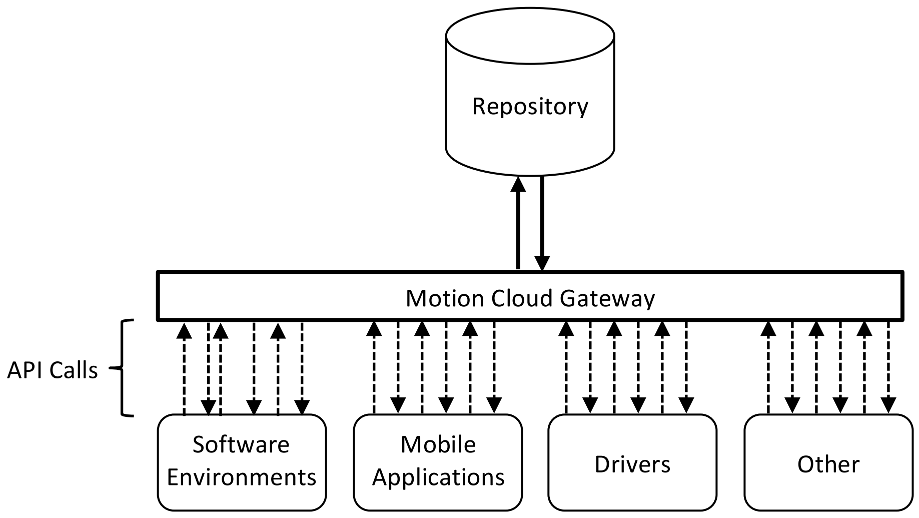


Figure 6-4: Motion Cloud gateway layer integrating third party software applications through API calls.

Each Motion Cloud API call is a PHP web service whose functionality can be identified by its title. For example, *channel_getlist* will return a list of channels, consisting of unique channel ids and names bound to an account and *channel_get* will return the contents of a single channel. Sending or retrieving data involves JavaScript Object Notation (JSON) packets and POST or GET HTTP requests.

The API has been implemented and tested using the software environment, mobile application and web portal. This section continues to discuss the profile, group, recording, channel and vector categories of API calls. In a similar way to the repository, the API architecture also forms a hierarchy whereby a call may itself access other calls. For example, making a request to obtain a recording's data will invoke nested channel and vector calls.

6.5.1 Profile API Calls

The first stage of interacting with the Motion Cloud API from a client-side application is to create a profile and authenticate. Figure 6-1 summarises the profile API calls, the necessary input parameters and the expected functionality. Example JSON packets for these calls can be found in Appendix C.1.

Table 6-1: Profile API calls.

API Call	Input Parameters	Functionality
profile_authenticate	Unique username and password.	Returns the user id if the password is correct and zero if it is not.
profile_create	Username, password, first name, last name and email address.	Creates a user profile with the given information and returns the profile's user id.
profile_get	Unique user id.	Returns a JSON packet containing the username, password, first name, last name and email address.
profile_update	User id, username, password, first name, last name and email address.	Updates a user profile with the given information.
profile_delete	Unique user id.	Deletes a profile object.
profile_check_username	Unique username	Checks whether the username is taken.

6.5.2 Group API Calls

Table 6-2 summarises the group API calls, the necessary input parameters and the expected functionality. As previously mentioned, groups are an optional part of the API designed to organise repository data. Example JSON packets for these calls can be found in Appendix C.2.

Table 6-2: Group API calls.

API Call	Input Parameters	Functionality
group_getlist	Unique user id.	Returns a JSON packet containing all list of groups bound to one account. The list consists of group unique ids, titles and descriptions.
group_create	Unique user id, title and description.	Creates a blank group object.
group_get	Unique group id.	Returns a JSON packet containing a hierarchy of recording objects containing channel and vector objects.
group_add	Unique group id and recording id.	Adds a recording to a group.
group_remove	Unique group id and recording id.	Removes a recording from a group.
group_delete	Unique group id.	Deletes a group.

6.5.3 Recording API Calls

The recording API is potentially the most used and important component of the gateway as every system, whether it's a single IMU or a whole BSN, uploads data as recording objects. Table 6-3 summarises the API calls. Example JSON packets for these calls can be found in Appendix C.3.

Table 6-3: Recording API calls.

API Call	Input Parameters	Functionality
recording_getlist_uid	Unique user id.	Returns a JSON packet containing all the unique recording ids, names and descriptions bound to one profile object.
recording_getlist_gid	Unique group id.	Returns a JSON packet containing all the unique recording ids, names and descriptions bound to one group object.
recording_create	JSON packet containing a title, description and hierarchy of channels objects.	Creates a blank recording object.
recording_get	Unique recording id.	Returns a JSON packet containing a hierarchy of channel and vector objects.
recording_update	JSON packet containing a title, description and hierarchy of channels objects.	Updates a recording with new channel and vector objects. Those objects are concatenated to the end of the recording.
recording_share	Unique user id of owner, username of receiver and unique recording id.	Bounds a recording object to the second user account. The recording will appear on both user accounts.
recording_clear	Unique recording	Deletes all the channel objects contained by the recording.
recording_delete	Unique recording id.	Deletes a recording object.

6.5.4 Channel API Calls

Table 6-4 summarises the channel category of API calls. Example JSON packets for these calls can be found in Appendix C.4.

Table 6-4: Channel API calls.

API Call	Input Parameters	Functionality
channel_getlist	Unique recording id.	Returns a JSON packet containing a list of channel titles and names.
channel_create	Unique recording id and channel name.	Creates a channel and adds it to a recording object.
channel_get	Unique channel id.	JSON packet containing channel id, name and a list of vector objects.
channel_update	JSON packet containing recording id and a list of vector objects.	Updates a channel with new vector objects.
channel_delete	Unique channel id.	Deletes a channel object.

6.5.5 Vector API Calls

As illustrated by Table 6-5, the API also allows software environments to access individual vector objects. While this category of calls is primarily used by the API itself, it provides developers with complete control of the repository data. Example JSON packets for these calls can be found in Appendix C.5.

Table 6-5: Vector API calls.

API Call	Input Parameters	Functionality
vector_gelist	Unique channel id.	Returns a JSON packet containing a list of vector ids.
vector_create	Unique channel id, yaw, pitch and roll values.	Creates a new vector object and adds it at the end of the specified channel object.
vector_get	Unique vector id.	JSON packet containing a unique vector id, yaw, pitch and roll values.
vector_update	Unique channel id, yaw, pitch and roll values.	Creates a vector and adds it to a recording object.
vector_delete	Unique vector id.	Deletes a vector object.

6.5.6 Stringing Together API Calls

The act of integrating the Motion Cloud API with a client-side application involves stringing together API calls in a specific order.

Figure 6-5 illustrates how a client-side application can upload a recording. The first step is to authenticate through the interface by entering a unique username and a secret password. The server-side controller takes those values and posts them to retrieve a unique profile id. The user id is then posted back to the server along with a JSON packet containing a recording's data. The Motion Cloud parses the packet and creates a recording object in the repository. The recording object is populated with channels and each channel is populated with a list of vectors. To finalize the process, the application retrieves a recording id for safekeeping and to access the data at a later stage.

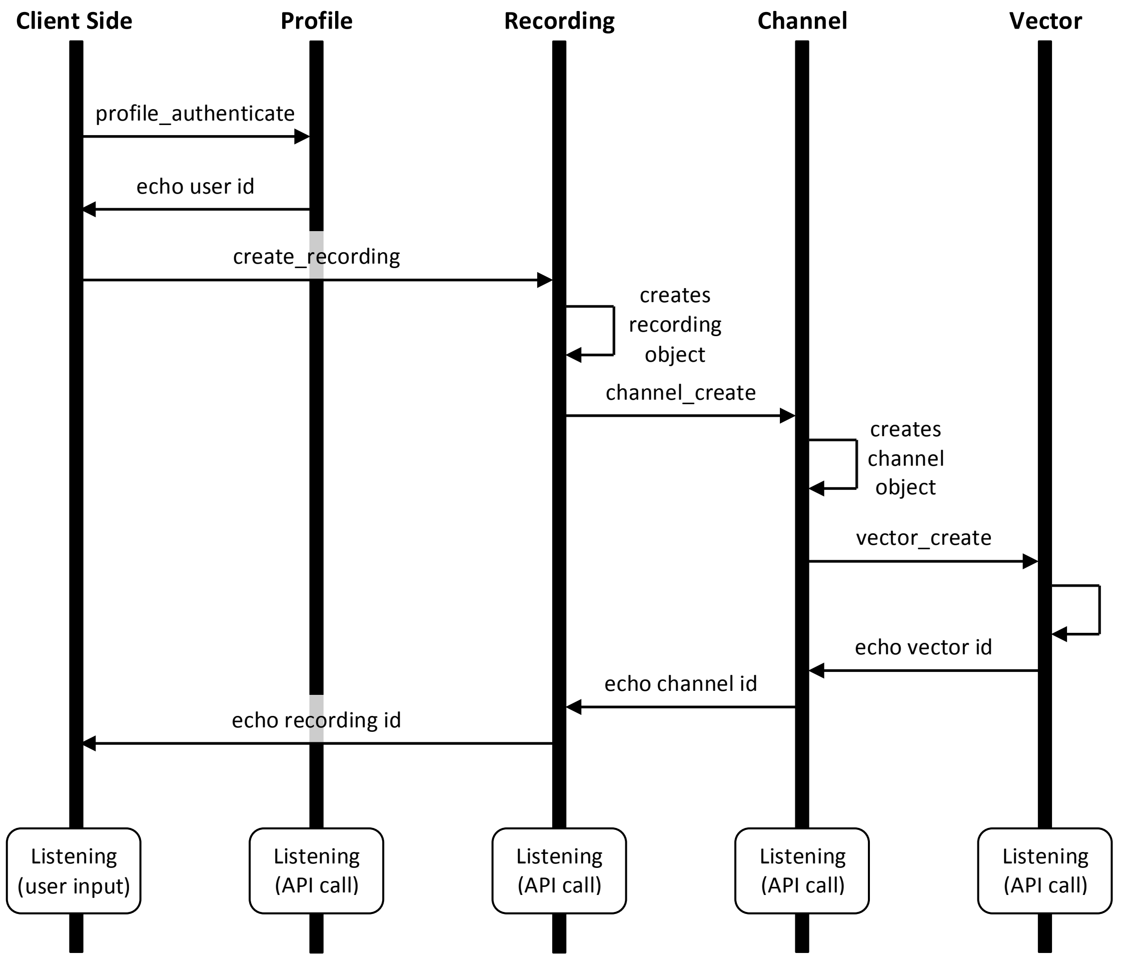


Figure 6-5: Sequence diagram illustrating how API calls can form a BSN.

Figure 6-6 shows the integration of the API with the Skeletrix mobile application by demonstrating how two interconnected smartphones can form one BSN. More specifically, this example demonstrates how one smartphone's data is communicated across the BSN to a second smartphone. This example assumes that the user has authenticated and chosen a recording on both devices.

Smartphone S_1 is tapped to initiate the BSN and subsequently modifies the recording status from 0 to 1. S_1 continues to update a channel's data with new angular readings. S_2 is listening for status updates and detects a change. Because the recording status is set to 1, S_2 begins recording by updating another channel's data with new angular readings. After the two channels are updated, both smartphones download a fresh copy of the recording and place it into local storage.

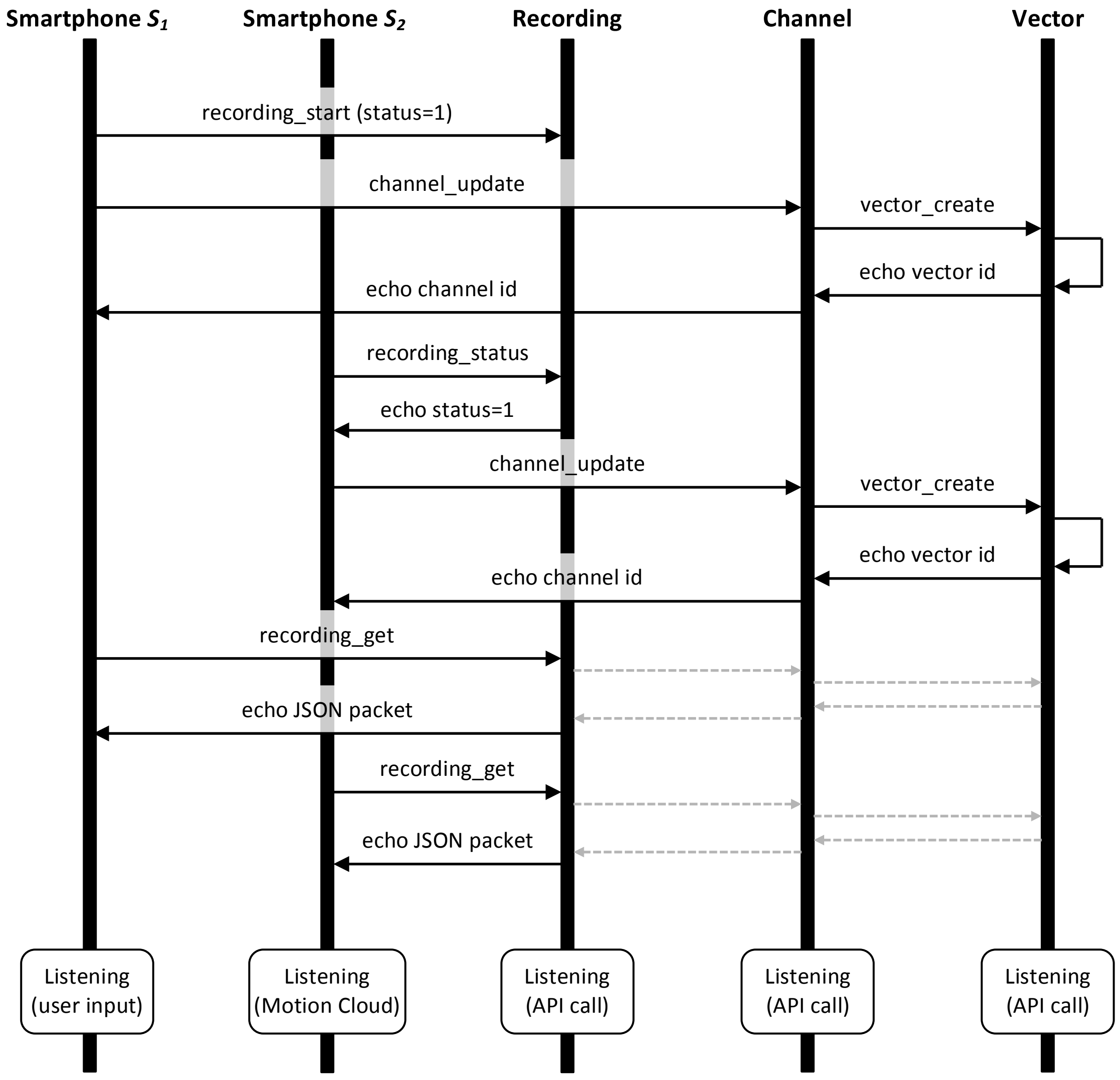


Figure 6-6: Uploading a recording object using the Motion Cloud API.

6.5.7 API Integration

The Motion Cloud API was integrated with the software environment, mobile application and web portal. Table 6-6 shows how the API was integrated within Skeletrix framework with the web portal, mobile application and software environment. Because the web portal and the repository are stored on the same server, the API functionality was absorbed into the web portal code to prevent the unnecessary JSON packet transfers and optimize the system.

Table 6-6: Framework API integration.

API Call	Web Portal	Mobile Application	Software Environment
profile_authenticate	**YES**	**YES**	**YES**
profile_create	**YES**	**YES**	N/A
profile_get	**YES**	N/A	N/A
profile_update	**YES**	N/A	N/A
profile_delete	**YES**	N/A	N/A
profile_check_username	**YES**	**YES**	N/A
group_getlist	**YES**	N/A	N/A
group_create	**YES**	N/A	N/A
group_get	**YES**	N/A	N/A
group_add	**YES**	N/A	N/A
group_remove	**YES**	N/A	N/A
group_delete	**YES**	N/A	N/A
recording_getlist_uid	**YES**	**YES**	**YES**
recording_getlist_gid	**YES**	N/A	N/A
recording_create	**YES**	N/A	N/A
recording_get	**YES**	**YES**	**YES**
recording_update	**YES**	**YES**	**YES**
recording_start	**YES**	**YES**	N/A
recording_stop	**YES**	**YES**	N/A
recording_status	**YES**	**YES**	N/A
recording_share	**YES**	N/A	N/A
recording_clear	**YES**	**YES**	N/A
recording_delete	**YES**	N/A	N/A
channel_getlist	**YES**	N/A	N/A
channel_create	**YES**	N/A	N/A
channel_get	**YES**	N/A	N/A
channel_update	**YES**	**YES**	N/A
channel_delete	**YES**	N/A	N/A
vector_gelist	**YES**	N/A	N/A
vector_create	**YES**	N/A	N/A
vector_get	**YES**	N/A	N/A
vector_update	**YES**	N/A	N/A
vector_delete	**YES**	N/A	N/A

6.6 Web Portal Design and Implementation

The Motion Cloud's data is made accessible to users through a web portal, in a format that is suitable for downstream applications (e.g. establishing a motion library to showcase and analyse recorded data, creating a database for activity tracker data, etc.). The web portal is an interface layer developed using the aforementioned API together with, HTML, JavaScript and PHP. This section discusses the development of the web portal in terms of user journey and functionality.

6.6.1 Use Cases

Figure 6-7 summarises all the functionality possible through the web portal. Users must register and authenticate to gain access to the dashboard. The dashboard allows user to open their profile, select recordings or groups of recordings. Once a recording is selected, the user can visualise, modify, export or share its contents with other users.

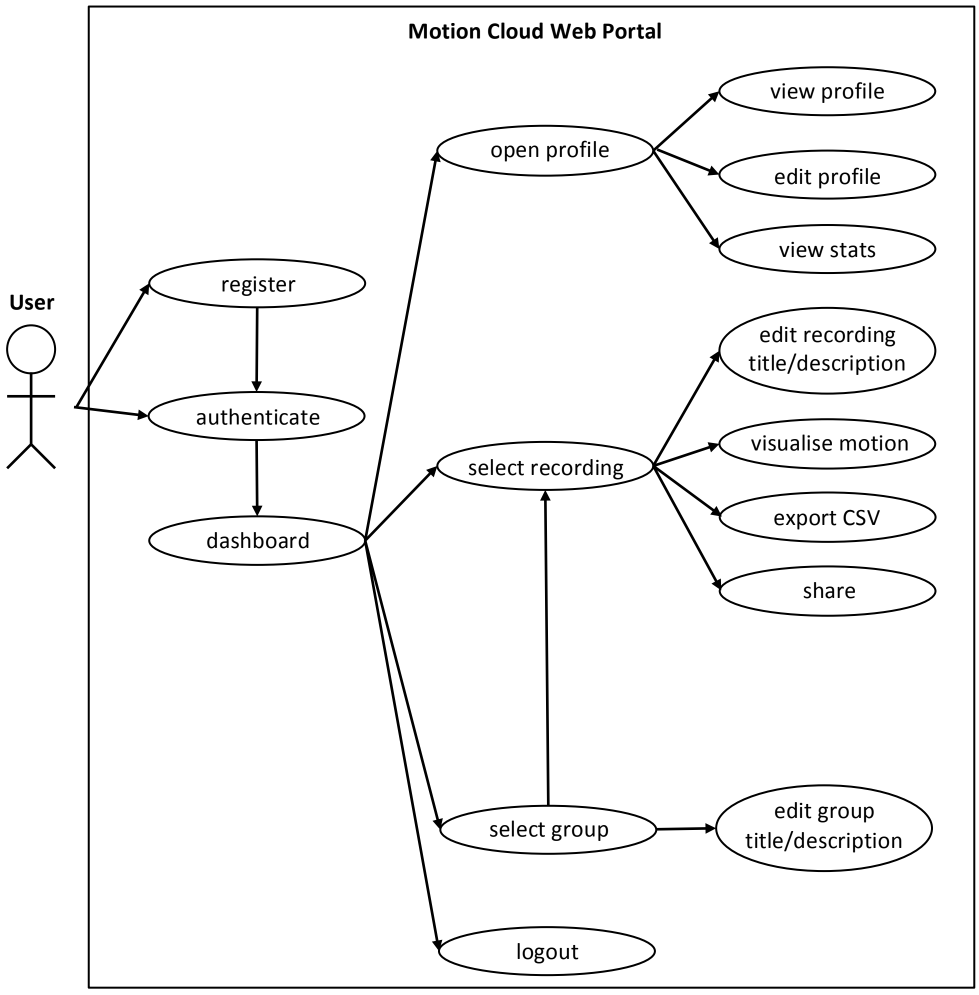


Figure 6-7: Motion Cloud web portal use case diagram.

6.6.2 Main Interface

As shown in Figure 6-8, the most elaborate interface of the web portal is the recording page as it provides most of the functionality. Aside from the header and footer, the recording interface can be divided into three sections: recording panel, BSN controller and channel visualizer.

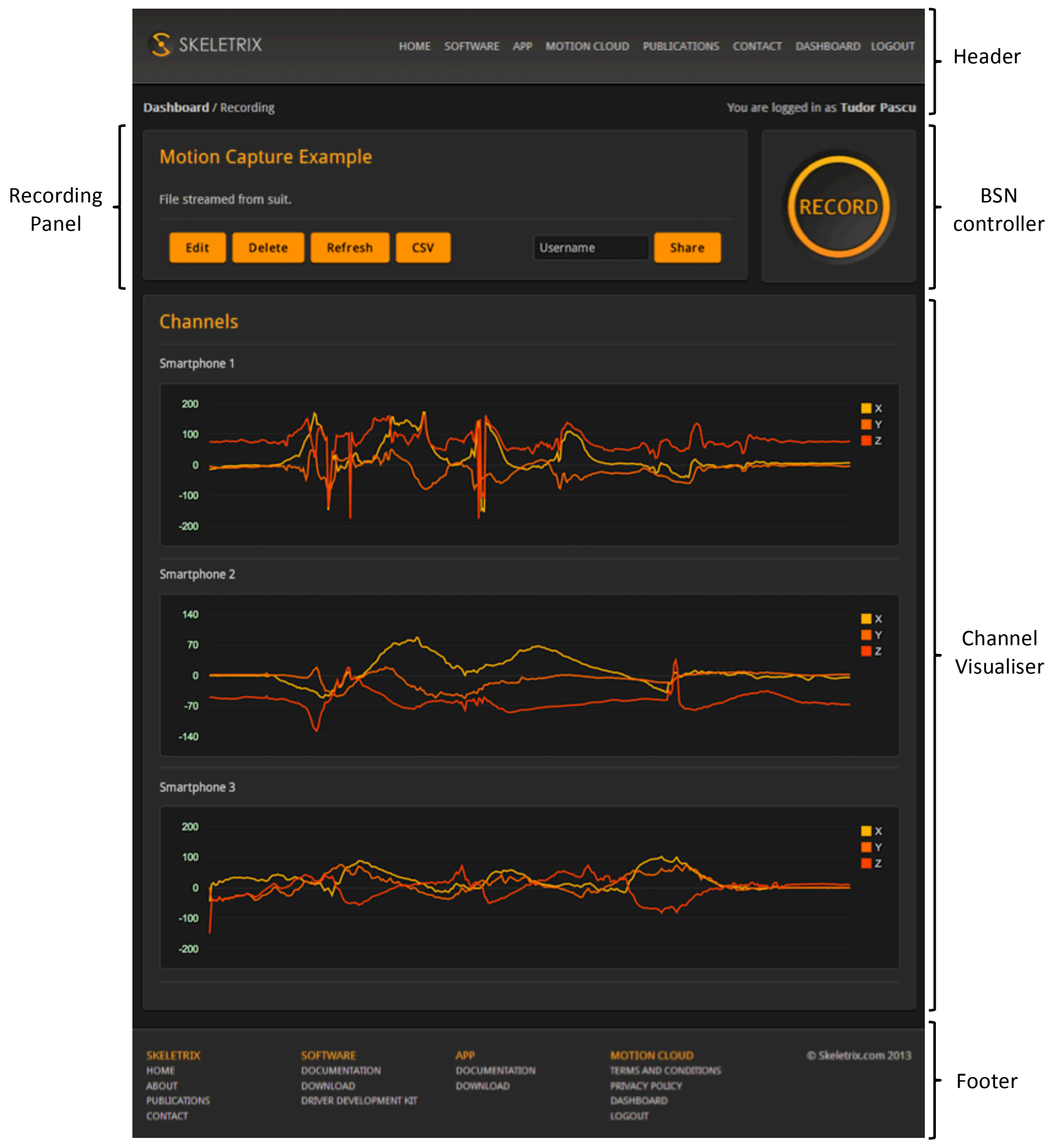

SKELETRIX
HOME SOFTWARE APP MOTION CLOUD PUBLICATIONS CONTACT DASHBOARD LOGOUT
Header
Dashboard / Recording
You are logged in as Tudor Pascu
Recording Panel
Motion Capture Example
File streamed from suit.
Edit
Delete
Refresh
CSV
Username
Share
RECORD
BSN controller
Channels
Smartphone 1
Smartphone 2
Smartphone 3
X
Y
Z
Channel Visualiser
SKELETRIX HOME ABOUT PUBLICATIONS CONTACT
SOFTWARE DOCUMENTATION DOWNLOAD DRIVER DEVELOPMENT KIT
APP DOCUMENTATION DOWNLOAD
MOTION CLOUD TERMS AND CONDITIONS PRIVACY POLICY DASHBOARD LOGOUT
© Skeletrix.com 2013
Footer

Figure 6-8: Motion Cloud web portal user interface.

The recording panel is located at the top of the screen. Each recording has a description and a title, which can be edited through the recording page. By editing the description, users can write explanatory text summarising the meaning of the motion data. The recording panel also allows users to delete, share or export the data in the

CSV formats. The share functionality allows several accounts to be linked to the same recording object.

The BSN controller allows users to remote control the recording process of an online BSN. In the context of smartphone-driven BSNs, clicking the record button will have the same effect as tapping the smartphone interface. All the smartphones connected to the recording will begin capturing motion. Like the mobile application interface, the record button changes from orange to green to give users a visual cue of the BSN status.

Using the Google Chart API [179], the channel visualizer creates a graph for every channel object belonging to the recording. Graphs allow for a preliminary interpretation the motion, before any 3D visualisations. Future iterations of the web portal may support Web Graphics Library (WebGL) [180], allowing for skeletal visualisations in the web browser.

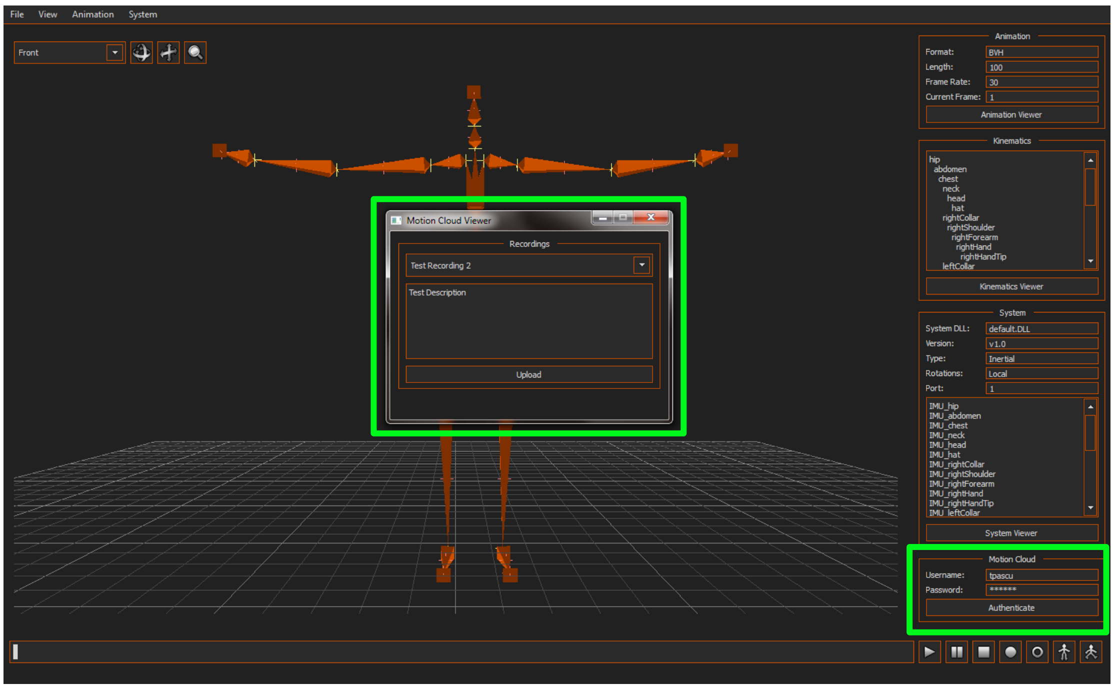
6.7 Motion Cloud Integration

The integration of the Motion Cloud has been demonstrated in the previous chapter in the context of smartphone-driven BSNs where the gateway API is used to transfer data between smartphones. However, the API plays a larger role throughout the framework as it facilitates the transfer of data between the Skeletrix software environment and the mobile application. This section discusses the integration of the Motion Cloud with the Skeletrix software environment through the development of the MotionCloudViewer.

The software environment was given Motion Cloud support through an additional viewer system, which resembles the structure of the AnimationViewer, KinematicsViewer and SystemViewer. The MotionCloudViewer allows users to extract data from sensors and upload it to the repository for safekeeping. This solution could be considered an alternative to motion capture file formats. Users can record motion data and upload it to the server. The motion data can be accessed or downloaded from the server at a later stage. Therefore, the server removes the need for any local storage and becomes a web-based substitute to motion capture file formats. This solution is important because it removes the need for local file storage by automating the process of soring and organising motion files. This solution saves users time and effort while simplifying the act of using a motion capture system and improving overall the motion capture workflow.

The MotionCloudViewer is closely interlinked with the animation model, which stores all the motion data. To upload data, the viewer grabs a copy of the animation model and iterates through the data to construct a JSON packet. That JSON packet is then posted to the server using a *QNetworkAccessManager*. *QNetworkReply* objects are used to receive messages from the server.

Figure 6-9 illustrates the MotionCloudViewer interfaces. A section is added on the main GUI that allows users to authenticate with the Motion Cloud. A successful authentication will open the MotionCloudViewer interface where users can chose to upload the whole recording to a recording object in the repository.



Test Recording 2

Test Description

Format: BVH
Length: 100
Frame Rate: 30
Current Frame: 1

hip
abdomen
chest
neck
head
hat
rightCollar
rightShoulder
rightForearm
rightHand
rightHandTip
leftCollar

System DLL: default.DLL
Version: v1.0
Type: Inertial
Rotations: Local
Port: 1

IMU_hip
IMU_abdomen
IMU_chest
IMU_neck
IMU_head
IMU_hat
IMU_rightCollar
IMU_rightShoulder
IMU_rightForearm
IMU_rightHand
IMU_rightHandTip
IMU_leftCollar

Username: tpascu
Password: ******

Figure 6-9: MotionCloudViewer adds Motion Cloud support to the Skeletrix software environment.

Future iterations of the MotionCloudViewer will be focused on streaming data from systems. For example, motion could be visualised on a smartphone as it is being recorded by a full-body motion capture suit. Mobile computing devices could be used during recording sessions along with inertial motion capture systems to monitor and visualise the motion data.

6.8 Conclusions

Fundamentally, the Motion Cloud is a large database designed to store inertial data produced by software environments (e.g. the Skeletrix software environment), mobile applications (e.g. the Skeletrix mobile application) and other third-party software applications. The Motion Cloud is used to demonstrate an implementation of web technologies that is designed specifically for the field of inertial motion capture and BSNs. The proposed solution is extendable and versatile, to meet the requirements of a wide range of sensors. Within the scope of the Skeletrix framework, the Motion Cloud is used to form a bridge between the developments presented in the previous chapters, namely the software environment and mobile application.

This chapter begins with a requirement specification that puts emphasis on the design of the three main components that constitute Motion Cloud, namely the repository, gateway and web portal.

The repository is a centralised database for inertial motion capture data originating from a single sensor or a more complex BSN. Unlike other motion databases, the Motion Cloud repository parses the data and divides it into database objects. The data model consists of a hierarchy of profile, recording, group, channel and vector objects. This unique approach allows for data streaming, online visualisations of motion, online data manipulation and other functionalities that are not possible with traditional motion libraries. Such functionality is possible because the data is parsed by the gateway and stored by the repository as objects rather than motion files.

The gateway contains an API layer allowing the integration of third-party software environments. The API consists of web services for manipulating the contents of the repository. This chapter has presented the API documentation and provided an example of stringing together API calls to upload a BSN’s data. Like the repository, the API architecture also forms a hierarchy whereby a call may itself access other calls.

The web portal is an interface layer for the Motion Cloud allowing users to visualise, modify and retrieve inertial motion capture data. Its purpose is to make data accessible to researches in a format that is suitable for experimental research. Additionally, the web portal is used to demonstrate an implementation of the API.

6.8.1 Application Areas

This section showcases potential usability scenarios for the Motion Cloud that illustrate its versatility and wide range of application areas. The mobile application has already demonstrated how the Motion Cloud can be used as a multiplexer to construct online BSNs. Let's consider four additional usability scenarios: motion library, prototyping environment, motion gateway and activity database.

Scenario 1: Motion Library

Most motion capture software and hardware developers provide libraries of motion, containing pre-recorded samples of motion, to their clients to showcase the accuracy of their systems. These libraries only allow users to upload or download files. There are no solutions for modifying or visualising data in a web browser. The current version of the Motion Cloud can be used as a motion library while providing a more detailed methodology for interacting with the data. Because each rotational value is stored as an entry in the database, it can be accessed with ease through a web portal or through web services. The web portal could be developed to contain motion analysis and cleanup tools, thus allowing users to inspect or edit the data quickly. This is not possible with existing motion libraries that contain the motion data uploaded as files. While the web portal provides an example interface layer for the repository, third-party software developers can build better software or web-based interfaces that suit their specific needs.

Scenario 2: Prototyping Environment

While most IMU manufacturers provide open-source drivers for their products, researchers have to develop or integrate software for data visualisations and storage. Inertial data can be uploaded directly to the Motion Cloud from the driver. The result can be stored, visualised or accessed in a convenient format without any additional software development. To demonstrate this concept, a standalone driver was developed for the Razor IMU. A standalone driver is a small program that sits between an application and a hardware device. In this context, the application is the Motion Cloud and the device is a Razor IMU powered by an Arduino board and connected to the computer through a USB cable. Its implementation is derived from the driver presented in Appendix B. The standalone driver is a small executable that runs in the background, without an interface, and streams data to the server as buffers containing several frames of motion each.

Scenario 3: Motion Gateway

A unique property of the Motion Cloud is the gateway, which allows users to send motion data between software environments, mobile applications, BSN nodes, etc. Effectively, the Motion Cloud becomes an online substitute for a multiplexer. This aspect has been demonstrated in the previous chapter in the context of online smartphone-driven BSNs.

Scenario 4: Activity Database

Activity tracking is an important application area for inertial sensing devices such as pedometers. The Motion Cloud can be used as a database for activity data (e.g. from a FitBit [173], Jawbone [174], etc.). That activity data can be stored in several formats. First, processed activity data can be stored in the form of calories burnt, steps taken and distance travelled. Second, raw activity data can be stored as rotations or gravitational accelerations to be used by processed at a later stage. In both scenarios, the Motion Cloud has the potential to be used as an activity database.

CHAPTER SEVEN

7 Conclusions

7.1 Summary

This chapter concludes the work presented throughout this thesis by summarising and discussing the four main research bodies: inertial motion capture software environments, constructing inertial body sensor networks (BSN), sensing through mobile computing technologies and the Motion Cloud, a repository and gateway for inertial motion data. Potential extensions and plans for future development are also discussed. To summarise, this thesis has contributed to the field of inertial motion capture and BSNs through four developments, each presenting unique solutions to common problems. When combined, these developments produce a framework for constructing BSNs, which extracts motion data from inertial hardware and makes it accessible to downstream applications through the Motion Cloud.

7.2 Inertial Motion Capture Software Environments

The Skeletrix software environment published in Pascu et al. [17] [18] is developed in an attempt to define an improved motion capture workflow by analysing the list of procedures involved in extracting and computing inertial motion capture data. Notably, the software environment is not a motion-editing tool for character animation and does not replicate the core functionality found in other software applications. Instead, the software environment is presented as a tool for researchers interested in studying or developing BSNs. To summarise, the Skeletrix software environment's first purpose is to provide a suitable experimentation environment, accompanied by programming scaffolding and a driver development kit, for users interested in integrating inertial BSNs or singular inertial measurement units (IMU) that enclose gyroscopes, accelerometers and magnetometers. The software environment architecture consists of three major core components: kinematic, animation and system engine. The kinematic model is used to construct a kinematic model that represents a virtual interpretation of the skeletal rig. The animation model stores recorded data and applies it to the kinematic model to animate skeletal rigs. The system model creates a virtual representation of the BSN at the software level, thus allowing users to develop the inner workings of the hardware. In accordance with the concept of transparency, each model is given a viewer, which is an interface layer for viewing the model's inner workings. The visualisation engine uses the OpenGL graphics pipeline to render 3D visualisations of the recorded motion.

The standardization of file formats has become an important requirement for developing new software environments [131]. The lack of a standard implies an industry that has too many incomplete solutions competing for software support. For this reason, introducing an entirely new format can be counterintuitive as developers are not eager to learn or implement new file systems. The proposed solution is to extend an existing format that developers are already familiar with. Biovision Hierarchy Extended (BVHE) lies at the core of the Skeletrix software environment and aims to tighten the relationship between software and inertial motion capture hardware. Users can configure the BSN by simply loading a BVHE file.

The Skeletrix software environment introduces the concept of a driver development kit (DDK) as a bridging architecture between hardware and software whereby driver modules can be developed for single IMUs or BSNs. A driver is a self-contained dynamic-link library (DLL) enclosing the code required to extract and parse the stream of motion data from the sensors. A driver must be built in accordance to a specification that allows the Skeletrix software environment to establish a valid connection. The first step is to connect the functions of the system model with the functions of the drivers, thus allowing the system model to control the actions of the driver. The second step focuses on hardware handshaking and is specific to each system. If the bridging is successful, data will begin to stream from the hardware and will display in the SystemViewer. Several drivers could be bridged simultaneously to stream data from several heterogeneous systems in real-time. As the implementation of the heterogeneous BSN demonstrates, the DDK facilitates a more flexible methodology for constructing BSNs and obtaining data from hardware.

The software environment presents a novel approach for computing the correct anchor of kinematic models during gate, which is based on the lowest-point algorithm. The lowest-anchor centre of mass algorithm (LACOMA) (published in Pascu et al. [18]) works using a weight model, which is constructed by adding physics, such as weight and weight distribution, to the kinematic model. The algorithm calculates the body's musculoskeletal axis of balance to determine the supporting anchor of the body. This solution prevents problems caused by motion performers dragging their feet, kneeling, crouching, etc.

To conclude, the Skeletrix software environment represents a tool for developing inertial motion capture systems. It provides a layer of software between the user and the hardware. This concept is demonstrated in Chapter 4 through the integration and development of IMUs. The fundamental architecture of the software environment is further used to develop the mobile application presented in Chapter 5.

7.3 Constructing Inertial Body Sensor Networks

Connecting sensors to the Skeletrix software environment through two case studies has been important to create a source of motion data and to demonstrate the functionality of the DDK and BVHE. This research work led to the development of the MTDS (published in Pascu et al. [18]), which is a new prototype-level inertial motion capture system.

Current BSNs are developed to be software-centric whereby all the data is extracted from the hardware, in its raw form, and sent to the computer for processing. Raw data is larger in size and therefore the BSN needs to communicate larger messages between its nodes and to the computer. For example, software-centric IMUs containing gyroscopes, accelerometers and magnetometers output nine degrees of freedom. On the other hand, hardware-centric IMUs output fused data corresponding to three degrees of freedom. The Motion Tracking Development System (MTDS) demonstrates that hardware-to-hardware and hardware-computer intercommunications can be reduced if microcontrollers are programmed to process data at the hardware-level. Hardware-centricity is exploited in MTDS system through two hardware developments: a central multiplexer connecting a constellation of IMU nodes. Only four batteries can power the suit because the hardware is designed efficiently by using low-power microcontrollers.

7.4 Sensing Through Mobile Computing Technologies

Modern smartphones embed a variety of sensors such as gyroscopes, accelerometers and magnetometers. This combination of sensors can also be found in most IMUs. Modern smartphones meet the computational requirements to sense and compute motion data. The proposed mobile application (Pascu et al. [15] [16] [19]) draws a parallel between mobile computing technologies and inertial motion capture with the goal of demonstrating the concept of smartphone-driven BSNs. The BSN is constructed using a centralised web server that replicates the functionality of a multiplexer. The conceptualisation of a smartphone-driven BSN is achieved through: motion processing, event triggers, data streaming protocols and data synchronization.

The smartphone-driven BSN is evaluated in several contexts. To begin with, two smartphones are moved on a flat surface to capture and simulate the articulated kinematic motion of a hand wave. This experiment is taken further as three smartphones are placed on the body to record the articulated motion of a hand wave gesture. The results show that in the context of inertial character animation, the smartphone sensors show potential in terms of performance. The smartphone's sensors are becoming sufficiently accurate to be deployed on bespoke circuit boards to create IMUs for BSNs, which can be used to capture kinematic human motion.

This thesis continues to investigate a second application area, activity tracking. Two smartphones are networked to evaluate the difference between sedentary and active behaviour. There is a potential for the mobile application to be repurposed as an activity tracker, which produces a rough estimation of steps taken, calories burnt and distance travelled. Additional usability scenarios, which illustrate the versatility of the mobile application, are also discussed: medical science, emergency responses, road and traffic condition monitoring.

7.5 Motion Cloud: A Repository and Gateway for Inertial Motion Data

The Motion Cloud is developed in an attempt to extend the motion capture workflow with web technologies. Throughout this thesis, the Motion Cloud has elaborated the concept of storing, organising and accessing that data through extendable database models and multipurpose web services. The unique Motion Cloud architecture is designed to accommodate any configuration of sensors, from single sensors to full-body sensor networks streaming data online. The Motion Cloud is the combined result of three main developments: the centralised motion repository, the gateway API and the web portal interface layer.

The Motion Cloud's centralised repository is a large database designed to store inertial data produced by software environments, mobile applications and drivers. While most motion capture hardware developers provide online databases for storing motion files, this repository differentiates itself from existing solutions through an object-oriented design. Motion data is parsed and divided into objects that are subsequently stored in a database hierarchy. The hierarchy structure simulates the configuration of a kinematic model.

The Motion Cloud's gateway interconnects the Skeletrix software environment and mobile application presented in this thesis to form one framework. As a result, the gateway can also be used to transfer data between platforms. For example, motion recorded using a desktop computer and running the Skeletrix software environment can be uploaded, using the MotionCloudViewer, to the Motion Cloud and downloaded using the mobile application.

The goal of this research work is to change the way users store and transfer motion data between computers, thus eliminating the need for local storage of data and motion capture file formats. The gateway is also presented as an intercommunications mechanism for developing online wireless BSNs. As a result, data can be streamed by each BSN node and downloaded by the whole network, a concept that is evaluated using mobile computing technologies. To make the Motion Cloud relevant beyond the scope of the Skeletrix framework, the gateway provides an API allowing third-party software applications to upload, download and stream data to and from the repository.

The Motion Cloud's web portal is an interface layer for the repository and the gateway. It provides tools for visualising, creating, modifying, deleting and exporting motion data in a format that is suitable for downstream applications. For example, the web portal could be used by healthcare workers to evaluate a patient's response to treatments as discussed in Pascu et al. [15] [16]. Four additional usability scenarios, which illustrate the versatility of the Motion Cloud, are also discussed: motion library, prototyping environment, motion gateway and activity database.

The Motion Cloud also blurs the boundaries of BSNs by integrating other inertial sensing devices, such as commercial pedometers (e.g. FitBit [173], Jawbone Up [174]), in the Skeletrix framework. Therefore, the Motion Cloud has the potential to become an Internet of Things [34] platform through which all sensors are unified to produce data that is stored in one centralised repository.

7.6 Towards a More Efficient Motion Capture Workflow

Without the developments presented in this research, the traditional workflow for a typical motion capture suit is limited. With the developments presented in this thesis, the workflow is improved and extended with new technologies. This section presents two contrasting usability scenarios.

Typical Inertial Motion Capture Scenario:

> A motion performer and a system operator use a motion capture suit to record full-body motion for the period of one hour. The purpose of the recording is to analyse human-environment interaction throughout urban households. The performer is dressed in the hardware with the help of the operator. This process takes approximately half an hour as nineteen IMUs and their interconnecting cables must be secured firmly on the body in a particular configuration (in accordance to the instruction manual). The suit is switched on and the network handshaking begins whereby the computer interrogates all sensor nodes individually. The user is asked to stand next to a reference object, which is used by the operator to adjust the onscreen kinematics to match the motion performer's bodily proportions. The motion performer faces north and the magnetometers are zeroed. Performing the T-pose, which involves standing straight with both arms extended laterally away from the body, compensates the postural difference. The suit begins recording if all these steps are performed successfully. While navigating the household environment, sensors may switch off due to loose or damaged connectors. Additionally, the chance of the suit losing wireless signal due to distance or environment obstructions is high. A single sensor disconnecting implies a complete system restart, which involves repeating the aforementioned procedures. Once the recording session has completed successfully, the operator takes the data and applies a set of filters and data cleaning algorithms. The result is clean motion that can be used for scientific analysis and is stored using one of several file formats.

Through the Skeletrix framework, the motion capture workflow (the series of procedures required to initiate a BSN that are showcased in the aforementioned scenario) is simplified or automated. To begin a recording, users simply load a BVHE file and perform the onscreen motion to calibrate. Hardware integration is made flexible so that users can decide the configuration of their system such as specifying the number and sensors and kinematic model. As demonstrated through the development of the DDK, hardware can be connected to the computer using DLL-enclosed drivers. The DDK is configured using the BVHE file format. As demonstrated by LACOMA, anchor selection can be improved by supplementing the kinematic model with weight models. Better anchor selection has the potential to improve the accuracy of foot placement estimation and subsequently the accuracy of dead reckoning. Using the Skeletrix framework, BSNs can be developed to be more hardware-centric whereby software functionality is achieved at the hardware level rather than software. As shown in the development of MTDS, embedding microcontrollers to process motion at the multiplexer and IMU levels, with carefully thought-out firmware, can solve this problem. The IMUs firmware can be designed so that the sensors restart autonomously, thus making the BSN more modular. Modularity was a design requirement as it improves versatility and makes the BSN applicable to a wider range of application areas.

For situations where bespoke BSNs are not required (e.g. small experiments requiring few sensors), smartphones can be used to sense inertial motion capture. This thesis demonstrates how the smartphone, the most ubiquitous type of wearable computing, can be integrated into an inertial motion capture framework. Smartphones also have the potential to help BSN development by becoming test beds or prototyping environments for inertial motion capture systems.

But is there a better methodology for storing, organising and transferring inertial data other than motion capture files once the motion data is obtained from hardware? As demonstrated by the Motion Cloud, web technologies can be used to extend the motion capture workflow and provide users with a better solution for storing, transferring and visualising motion capture data. That data is then made available for downstream applications through a unique and innovative API.

The following scenario puts the research work presented in this thesis into context.

Proposed Scenario:

> A motion performer uses an MTDS suit to record motion for the period of one hour. The purpose of the recording is to analyse human-environment interaction throughout urban households. The performer puts on the hardware, a process that takes minutes to complete because the suit is lightweight. The performer opens the Skeletrix software environments, loads a BVHE file that is suited to their experiment and begins recording. The suit initiates itself, starts recording and the user mimics the onscreen motion to calibrate. If a sensor disconnects during the recording, that specific sensor reboots and joins the BSN automatically. Once the recording session is complete, the software environment uploads the motion data to the Motion Cloud for further analysis. That motion data can later be visualised using web or mobile computing technologies.

7.7 Future Work

The research presented in this thesis showcases several prototypes that have the potential to be developed further. While those prototypes showcase a variety of concepts, there is room for further improvements and new functionality. The research work presented in this thesis is taken forward by two research project proposals for funding that are focused on Parkinson's disease research and digital tool chains for inertial motion capture. This section discusses potential extension for the Skeletrix software environment, mobile application, Motion Cloud and MTDS suit.

7.7.1 Skeletrix Software Environment Extensions

The Skeletrix software environment has demonstrated the benefits of using a motion capture file format that integrates system configuration data along with kinematic definitions and motion data. Although the semantics of BVHE contain all the information required to run a motion capture system, the syntax can be difficult to parse. In accordance to the premise of creating a file format that users are already familiar with, the Biovision Hierarchy (BVH) syntax was used as a starting point to provide users with a file type that they already understand. The development of new software that uses this format implies writing complicated parsers. A simpler and more efficient approach would be to utilise eXtensive Markup Language (XML) in a similar way to Motion Capture Markup Language (MCML) [131] or JavaScript Object Notation (JSON). JSON is currently the most efficient way of storing data while XML is more intuitive.

LACOMA computes motion in the horizontal plane based on foot placement estimation. However, there are many situations where the third axis is also required to determine vertical displacement. For example, the act of running implies "walking" where both feet detach from the ground plane during each consecutive gait cycle. It would be beneficial to add more physics to LACOMA so that users can perform gestures, such as jumping, where the kinematic model detaches entirely from the ground plane. However, this approach implies developing a basic physics engine and is beyond the scope of developing a solution for dead reckoning.

7.7.2 Skeletrix Mobile Application Extensions

The Skeletrix mobile application has demonstrated that inertial motion capture is possible using smartphones connected to the Internet. The resulting BSN is omnidirectional in the sense that every smartphone node communicates data to the entire network. The implementation of this solution is achieved using server-polling techniques. One smartphone uploads its data while the others constantly interrogate the server for updates. The server does not have the ability to send a message directly to a smartphone. A possible extension is to create a server controller that is abled to actively send data to the BSN nodes. This approach would decrease latencies and mobile data usage.

7.7.3 Motion Cloud Extensions

The Motion Cloud has demonstrated how web technologies can be used to enhance inertial motion capture in two ways. First, web technologies can extend the motion capture workflow to allow for a better methodology for storing and trading motion capture data. Second, web technologies can become an intercommunication mechanism for IMUs, allowing for more flexible wireless BSNs. There are other unexplored methods in which web technologies can become useful. A possible extension for the Motion Cloud is a social media module that brings together communities of developers interested in sharing knowledge.

The current data model is designed to accommodate only three-valued vectors such as angular readings or positional offsets. While the repository is designed specifically for inertial motion capture technologies, the data model can be used to store data from other mediums such as exoskeleton suits or optical sensors. All character motion capture technologies eventually produce animated kinematic models. Although the API requirement specification has taken into account those technologies, further research is required to develop the Motion Cloud in those contexts.

The web portal makes motion capture data, obtained from hardware, more accessible to users. The only method for data visualisation data is through graphs or in a tabular format. Web Graphics Library (WebGL) [180] is a JavaScript library for 3D visualisations based on OpenGL ES 2.0 that makes use of the computer's Graphics Processing Unit (GPU). A possible extension for the web portal is to visualise kinematic motion as demonstrated by the software environment and mobile application.

While the gateway API has been shown to work with the software environment and the mobile application, it has not been evaluated with commercial software. Software applications such as MotionBuilder [133] and Blender [137] allow users to develop plugins or modify the source code. A potential extension is to integrate the Motion Cloud with other software applications outside the Skeletrix framework.

7.7.4 MTDS Extensions

The MTDS upper body suit was developed as a prototype using consumer-level sensors and microcontrollers. Sensor fusion is not possible because the suit does not implement magnetometers. As a result, the sensors have a significant amount of drift whereby the system can only be used for short periods of time. A possible extension is to redevelop the printed circuit boards to accommodate magnetometers. Alternatively, modern sensors include the gyroscope, accelerometer, magnetometer and small microcontrollers in the same chip capsule. Integrating a modern sensor chip would minimize drift and improve overall accuracy.

In an attempt to improve power consumption, both the MTDS multiplexer and IMU use 8-bit microcontrollers to compute and transfer motion. However, the values produced by the sensors are floating-point numbers, which are inefficient to compute on 8-bit microcontrollers. As a result, a possible extension to the MTDS architecture is to upgrade the hardware with 32 bit ARM microcontrollers (e.g. Cortex A50 Series microcontrollers [181]). This added computational capabilities would allow for an even more autonomous motion capture system.

References

[1] Calvert, T.W., Chapman, J., Patla, A., "Aspects of the Kinematic Simulation of Human Movement", *Proceedings of IEEE Computer Graphics and Applications*, vol. 2(9), pp. 41-48, 1982.

[2] Raskar, R., Nii, H., DeDecker, B., Hashimoto, Y, Summet, J., Moore, D., Zhao, Y., Westhues, J., Dietz, P., Barnwell, J., Nayar, S., Inami, M., Bekaert, P., Noland, M., Branzoi, V., Bruns, E., "Prakash: Lighting Aware Motion Capture Using Photosensing Markers and Multiplexed Illuminators", *ACM Transactions on Graphics*, vol. 26(3), pp. 36, 2007.

[3] Roetenberg, D., Luinge, H.J., Baten, C.T.M., Veltink, P.H., "Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 13(3), pp. 395-405, 2005.

[4] Roetenberg, D., Luinge, H., and Veltink, P, "Inertial and magnetic sensing of human movement near ferromagnetic materials", *Proceedings of 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 268-269, 2003.

[5] Lander, J., "Working With Motion Capture File Formats", *Game Developer*, January 1998, Available from: http://www.darwin3d.com/gamedev/articles/col0198.pdf (Cited: 17/09/2013)

[6] Meredith, M., Maddock, S., "Motion Capture File Formats Explained", 2001, Available from: http://www.dcs.shef.ac.uk/intranet/research/public/resmes/CS0111.pdf (Cited: 17/09/2013)

[7] Jung, M., Fischer, R., Gleicher, M., Thingvold, J., "Motion Capture and Editing: Bridging Principles and Practices", September 2005, ISBN: 978-1568810867.

[8] Bodenheimer, B., Rose, C., Rosenthal, S., Pella, J., "The Process of Motion Capture: Dealing with the Data", *Computer Animation and Simulation*, 1997.

[9] Maestri, G., "Digital Character Animation", *New Riders Publishing*, 1996.

[10] A.D. Young, "From Posture to Motion: The Challenge for Real Time Wireless Inertial Motion Capture", *Proceedings of 5th International Conference on Body Area Networks*, 2010.

[11] Schepers, H.M., van Assledonk, E.H., Baten, C.T., Veltink, P.H., "Ambulatory Estimation of Foot Placement During Walking Using Inertial Sensors", *Journal of Biomechanics*, vol. 43(16), pp. 3138-3143, 2010.

[12] Rebula, J.R., Ojeda, L.V., Adamczyk, P.G., Kuo, A.D., "Measurement of Foot Placement and its Variability With Inertial Sensors", *Gait Posture*, Vol. 38(4), pp. 974-980, 2013.

[13] Fischer, C., Muthukrishnan, K., Hazas, M., Gellersen, H., "Ultrasound-Aided Pedestrian Dead Reckoning for Indoor Navigation", *Proceedings of the 1st ACM International Workshop on Mobile Entity Localization and Tracking in GPSless Environments (MELT08)*, pp. 31-36, 2008.

[14] Krüger, A., Edelmann-Nusser, J., "Application of a Full Body Inertial Measurement System in Alpine Skiing: A Comparison with an Optical Video Based System", *Journal of Applied Biomechanics*, vol. 26(4), pp. 516-521, 2010.

[15] Pascu, T., Patoli, M.Z., Barker, L., Beloff, N., White, M., "Ambient Health Monitoring: The Smartphone as a Body Sensor Network Component", *Proceedings of Innovation in Medicine and Healthcare (INMED)*, pp. 62-65, Piraeus, Greece, July 2013.

[16] Pascu, T., Patoli, M.Z., Barker, L., Beloff, N., White, M., "Ambient Health Monitoring: The Smartphone as a Body Sensor Network Component", *The Journal of Innovation Impact*, vol. 6(1), pp. 62-65, 2013.

[17] Pascu, T., Patoli, M.Z., White, W., "Unifying Software and Hardware-Centric Inertial Measurement Units in Body Sensor Networks", *Proceedings of 13th International Conference on Computer Graphics and Imaging (CGIM),* Innsbruck, Austria, February 2013.

[18] Pascu, T., Patoli, M.Z., White, W., "Improving Anchor Selection for Inertial Motion Capture Systems Through Weight Distribution Calculations", *Proceedings of 13th International Conference on Computer Graphics and Imaging (CGIM)*, Innsbruck, Austria, February 2013.

[19] Pascu, T., White, W., Patoli, M.Z., "Motion Capture and Activity Tracking Using Smartphone-Driven Online Body Sensor Networks", *Proceedings of 3rd International Conference on Innovative Computing Technology (INTECH)*, London, United Kingdom, August 2013.

[20] Technology Strategy Board, *Innovateuk*, Available from: http://www.innovateuk.org (Cited: 17/09/2013)

[21] Animazoo, Available from: http://www.animazoo.com (Cited: 17/09/2013)

[22] eMove, *MocapSuit*, Available from: http://www.mocapsuit.com/ (Cited: 17/09/2013)

[23] Patoli, M.Z., White, M., Gikon, M., "Real-time Online Digital Avatar with Lip Syncing and Facial Expressions", *Proceedings of 3rd IEEE International Conf. on Digital Game and Intelligent Toy Enhanced Learning*, Kaohsiung, Taiwan, 2010.

[24] Patoli, M.Z., Gikon, M., Newbury, P., White, M., "Real Time Online Motion Capture for Entertainment Applications", *Proceedings of 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, Kaohsiung, Taiwan, 2010.

[25] Dunn, S., Woolford, K., Barker, L., Taylor, M., Norman, S.J., White, M., "Motion in Place: a Case Study of Archaeological Reconstruction Using Motion Capture", *Proceedings of the 39th Conference on Computer Applications and Quantitative Methods in Archaeology*, Beijing, China, 2011.

[26] Woolford, K., Dunn, S., "Motion in place platform: virtual (re)presentations of Iron Age movement", *Leonardo Electronic Almanac*, 2010.

[27] Dunn, S., Hedges, M., Woolford, K., Barker, L., Norman, S.J., Taylar, M., White, M., Fulford, M., Clarke, A., Bailey, H., "Motion in place: archaeological reconstruction and motion capture", *Computer Applications and Quantitative Methods in Archaeology*, Beijing, China, April 2011.

[28] Arts and Humanities Research Council, Available from: http://www.ahrc.ac.uk/ (Cited: 17/09/2013)

[29] Archeology, *University of Reading*, Available from: http://www.reading.ac.uk/archaeology/ (Cited: 17/09/2013)

[30] White, M., Nee, C., Patoli, Z., Pascu, T., "Virtual Burglary Simulation", *Virtual Emergencies: Simulation Technology for Emergency Planning and Response*, Royal United Services Institute, London, United Kingdom, April 2011.

[31] Nee, C., White, M., Woolford, K., Pascu, T., Barker, L., "Examining Expertise In Residential Burglars: The Results Of A Pilot Study Using Innovative Technology", *68th Annual Meeting of the American Society of Criminology (ASC)*, Chicago IL, November 2012.

[32] Nee, C., White, M., Woolford, K., Pascu, T., Barker, L., Wainwright, L. "Examining expertise in burglars in a natural and a simulated environment", *Psychology Crime & Law*, Submitted January 2014.

[33] DigitalHub, *University of Sussex/American Express*, Available from: https://www.digitalhub.org.uk (Cited: 17/09/2013)

[34] Leon Barker, Martin White, Mairead Curran, Zeeshan Patoli, Benjamin Huggins, Tudor Pascu, Natalia Beloff, "Taxonomy for Internet of Things: Tools for Monitoring Personal Effects", *Proceedings of 4th International Conference on Pervasive and Embedded Computing and Communication Systems*, Lisbon, Portugal, January 2014.

[35] Rosenhahn, B., Klette, R., Metaxas, D., "Human Motion: Understanding, Modelling, Capture, and Animation", November 2010, ISBN: 978-9048177004.

[36] Dyer, S., Martin, J., Zulauf, J., "Motion Capture White Paper", December 1995, Available from: ftp://ftp.sgi.com/sgi/A%7CW/jam/mocap/MoCapWP_v2.0.html (Cited: 17/09/2013)

[37] Tobon, R., Restrepo, A., "The Mocap Book: A Practical Guide to the Art of Motion Capture Paperback", January 2010, ISBN: 978-0615293066.

[38] Vlasic, D., Adelsberger, R., Vannucci, G., Barnwell, J., Gross, M., Matusik, W., Popović, J., "Practical motion capture in everyday surroundings", *ACM Transactions on Graphics*, 2007.

[39] Albert, M., "Understanding Motion Capture for Computer Animation and Video Games", October 1999, ISBN: 978-0124906303.

[40] Lamouret, A., Panne, M., "Motion Synthesis By Example", *Proceedings of 7th International Workshop on Computer Animation and Simulation (EGGAS)*, Eurographics, 1996.

[41] MotionScan, *Depth Analysis*, Available from: http://depthanalysis.com/motionscan/ (Cited: 17/09/2013)

[42] "L.A. Noire Honored by Popular Mechanics with the Breakthrough Award", *Rockstar Games*, Available from: http://www.rockstargames.com/newswire/article/19211/la-noire-honored-by-popular-mechanics-with-the-breakthrough-awar.html (Cited: 18/09/2013)

[43] AnimaLive, *Animazoo*, Available from: http://www.animalive.com (Cited: 17/09/2013)

[44] Kinect, *Microsoft*, Available from: http://www.microsoft.com/en-us/kinectforwindows/ (Cited: 17/09/2013)

[45] Li, Y., "Hand gesture recognition using Kinect", *Proceedings of IEEE International Conference on Computer Science and Automation Engineering,* pp. 196-199, 2012.

[46] Patsadu, O., Nukoolkit, C., Watanapa, B., "Human gesture recognition using Kinect camera", *Proceedings of 9th International Conference on Computer Science and Software Engineering JCSSE (IEEE)*, pp. 28-32, 2012.

[47] Biswas, K.K., Basu, S.K., "Gesture recognition using Microsoft Kinect", *Proceedings of 5th International Conference on Automation, Robotics and Applications (ICARA)*, pp. 100-103, 2011.

[48] Das, S., Trutoiu, L., Murai, A., Alcindor, D., Oh, M., De la Torre, F., Hodgins, J., "Quantitative measurement of motor symptoms in Parkinson's disease: A study with full-body motion capture data", *Proceedings of Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6789-6792, 2011.

[49] Balint, G., Dezso, Z., Hunka, A., Lenti, J., Lovanyi, I., "Motion Capture vs. Traditional Medical Examinations", *Proceedings of 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications*, Tunisia, 2005.

[50] Tseng, Y.C., Wu, C.H., Wu, F.J., Huang, C.F., King, C.T., Lin, C.Y., Sheu, J.P., Chen, C.Y., Lo, C.Y., Yang, C.Q., Deng, C.W., "A Wireless Human Motion Capturing System for Home Rehabilitation", *Proceedings of 10th International Conference on Mobile Data Management Systems Services and Middleware*, 2009.

[51] Zabjek, K.F., Leroux, M.A., Coillard, C., Rivard, C.H., Prince, F., “Evaluation of segmental postural characteristics during quiet standing in control and Idiopathic Scoliosis patients”, *Clinical Biomechanics*, vol. 20(5), pp. 483-490, 2005.

[52] Menz, H.B., Lord, S.R., Fitzpatrick, R.C., “Acceleration patterns of the head and pelvis when walking are associated with risk of falling in community-dwelling older people”, *The Journals of Gerontology Series A Biological Sciences and Medical Sciences*, vol. 58(1), pp. 446-452, 2003.

[53] Menz, H.B., Lord, S.R., Fitzpatrick, R.C., “Age-related differences in walking stability”, *Age Ageing*, 32(2), pp. 137-142, March 2003.

[54] Vallis L.A., McFadyen B.J., “Children use different anticipatory control strategies than adults to circumvent an obstacle in the travel path”, *Experimental Brain Research*, vol. 167(1), pp. 119-127, 2005.

[55] Sadeghi, H., Prince, F., Zabjek, K.F., Labelle, H., “Simultaneous, bilateral, and three-dimensional gait analysis of elderly people without impairments”, *American Journal of Physical Medicine Rehabilitation Association of Academic Physiatrists*, vol. 83(2), pp. 112-123, 2004.

[56] Saboune, J., Charpillet, F., “Markerless Human Motion Capture for Gait Analysis”, *Computer*, 2005.

[57] Cloete, T., Scheffer, C., “Repeatability of an off-the-shelf, full body inertial motion capture system during clinical gait analysis”, *Proceedings of Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5125-5128, 2010.

[58] Ferrari, A., Cutti, A.G., Garofalo, P., Raggi, M., Heijboer, M., Cappello, A., Davalli, A., “First in vivo assessment of “Outwalk”: a novel protocol for clinical gait analysis based on inertial and magnetic sensors”, *Medical & Biological Engineering & Computing,* vol. 48(1), pp. 1-15, 2010.

[59] Cutti, A. G., Ferrari, A., Garofalo, P., Raggi, M., Cappello, A., Ferrari, A., “Outwalk: a protocol for clinical gait analysis based on inertial and magnetic sensors”, *Medical & Biological Engineering & Computing*, vol. 48(1), pp. 17-25, 2010.

[60] Kruger, A., Edelmann-Nusser, J., “Biomechanical analysis in freestyle snowboarding: application of a full-body inertial measurement system and a bilateral insole measurement system”, *Sports Technology*, vol. 2(1–2), pp. 17-23, 2009.

[61] Krüger, A., McAlpine, P., Borrani, P., Edelmann-Nusser, J., “Determination of three-dimensional joint loading within the lower extremities in snowboarding”, *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 226(2), pp. 170-175, 2011.

[62] Krüger, A., Edelmann-Nusser, J., “Application of a full body inertial measurement in alpine skiing: A comparison with an optical video based system”, *Journal of Applied Biomechanics*, vol. 26(4), pp. 516-521, 2010.

[63] Supej, M., “3D Measurements of alpine skiing with an inertial sensor motion capture suit and GNSS RTK system”, *Journal of Sports Sciences*, vol. 28(7), pp. 759-769, 2010.

[64] Matthew, F., Zengxi, P., David, S., Fazel, N., “Human motion capture sensors and analysis in robotics”, *Industrial Robot: An International Journal*, vol. 38(2), pp. 163-171, 2011.

[65] Field, M., Stirling, D., Naghdy, F., Zengxi, P., “Motion capture in robotics review”, *Proceedings of IEEE International Conference on Control and Automation (ICCA)*, pp. 1697-1702, 2009.

[66] Shon, A.P., Grochow, K., Rao, R.P.N, “Robotic imitation from human motion capture using Gaussian processes”, *Proceedings of 5th IEEE-RAS International Conference on Humanoid Robots*, pp. 129-134, 2005.

[67] Schaal, S., “Is imitation learning the route to humanoid robots?”, *Trends in Cognitive Sciences*, vol. 3(6), pp. 233-242, 1999.

[68] Demiris, J., Rougeaux, S., Hayes, G., Berthouze, L., Kuniyoshi, Y., “Deferred imitation of human head movements by an active stereo vision head”, *Proceedings of the 6th IEEE International Workshop on Robot Human Communication*, 1997.

[69] Kuniyoshi, Y., Inaba, M., Inoue, H., “Learning by watching: Extracting reusable task knowledge from visual observation of human performance”, *IEEE Transactions on Robotics and Automation*, pp. 799-822, 1994.

[70] Stevenson, J.M., Bossi, L.L., Bryant, J.T., Reid, S.A., Pelot, R.P., Morin, E.L, “A suite of objective biomechanical measurement tools for personal load carriage system assessment”, *Ergonomics*, vol. 47(11), pp. 1160-1179, 2004.

[71] Yebin, L., Stoll, C., Gall, J., Seidel, H.P., Theobalt, C., “Markerless motion capture of interacting characters using multi-view image segmentation”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 1249-1256, 2011.

[72] Vicon, *Vicon Motion Systems*, Available from: http://www.vicon.com (Cited: 25/09/2013)

[73] The Standard, *Vicon Motion Systems*, Available from: http://www.vicon.com/standard/ (Cited: 25/09/2013)

[74] A Hopping Success For Outdoor Capture, *Vicon Motion Systems*, Available from: http://www.vicon.com/Standard/Archive (Cited: 25/09/2013)

[75] Vicon Bonita, *Vicon Motion Systems*, Image source: http://www.vicon.com/System/Bonita (Cited: 25/09/2013)

[76] Optotrak Certus, *Northern Digital Inc.*, Available from: http://www.ndigital.com/lifesciences/certus-motioncapturesystem.php (Cited: 25/09/2013)

[77] Northern Digital Inc., Available from: http://www.ndigital.com/ (Cited: 25/09/2013)

[78] Burger H., Kuzelicki J., Marincek C., "Transition from sitting to standing after trans-femoral amputation", *Prosthetics and Orthotics International*, vol. 29(2), pp. 139-151, 2005.

[79] Bertram C. P., Lemay M., Stelmach G. E., "The effect of Parkinson's disease on the control of multi-segmental coordination", *Brain and Cognition*, vol. 57(1), pp. 16-20, 2005.

[80] Zupancic J., Bajd T., "Comparison of position repeatability of a human operator and an industrial manipulating robot", *Computers in Biology and Medicine*, vol. 28(4), pp. 415-421, 1998.

[81] Burgess-Limerick, R., Green, B., "Using multiple case studies in ergonomics: An example of pointing device use", *International Journal of Industrial Ergonomics*, vol. 26(3), pp. 381-388, 2000.

[82] Sadeghi H., Prince F., Zabjek K.F., Labelle H., "Simultaneous, bilateral, and three-dimensional gait analysis of elderly people without impairments", *American Journal of Physical Medicine & Rehabilitation*, vol. 83(2), pp. 112-123, 2004.

[83] Patla A.E., Tomescu S. S., Ishac M.G., "What visual information is used for navigation around obstacles in a cluttered environment?", *Canadian Journal of Physiology and Pharmacology*, vol. 82(8-9), pp. 682-692, 2004.

[84] Optotrak Certus, *Northern Digital Inc.*, Image source: http://www.ndigital.com/lifesciences/certus-stand.php (Cited: 26/09/2013)

[85] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A., "Real-time human pose recognition in parts from single depth images", *IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 1297-1304, 2011.

[86] Pittman, N., Forin, A., Criminisi, A., Shotton, J., Mahram, A., "Image Segmentation Using Hardware Forest Classifiers", *Proceedings of IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2013.

[87] Kinect SDK, *Microsoft*, Available from: http://www.microsoft.com/en-us/kinectforwindows/develop/ (Cited: 25/09/2013)

[88] Robotics Developer Studio, *Microsoft*, Available from: http://www.microsoft.com/robotics/ (Cited: 25/09/2013)

[89] Chang, C.Y., Lange, B., Zhang, M., Koenig, S., Requejo, P., Somboon, N., Sawchuk, A.A., Rizzo, A., "Towards Pervasive Physical Rehabilitation Using Microsoft Kinect", *Proceedings of 6th International Conference on Pervasive Computing Technologies for Healthcare*, 2012.

[90] Huang, Z., Nagata, A., Kanai-Pak, M., Maeda, J., Kitajima, Y., Nakamura, M., Aida, K., Kuwahara, N., Ogata, T., Ota, J., "Feedback-Based Self-training

System of Patient Transfer", *Digital Human Modelling and Applications in Health, Safety, Ergonomics, and Risk Management. Healthcare and Safety of the Environment and Transport*, pp. 197-203, 2013.

[91] Fern'ndez-Baena, A., Susin, A., Lligadas, X., "Biomechanical Validation of Upper-Body and Lower-Body Joint Movements of Kinect Motion Capture Data for Rehabilitation Treatments", *Proceedings of 4th International Conference on Intelligent Networking and Collaborative Systems (INCOS)*, pp. 656-661, 2012.

[92] Delpresto, J., Duan, C., Layiktez, L.M., Moju-Igbene, E.G., Wood, M.B., Beling, P.A., "Safe lifting: An adaptive training system for factory workers using the Microsoft Kinect", *Systems and Information Engineering Design Symposium (SIEDS)*, pp. 64-69, 2013.

[93] Ekelmann J., Butka B., "Kinect Controlled Electro-Mechanical Skeleton", Proceedings of IEEE IEEE SoutheastCon, pp. 1-5, 2012.

[94] El-laithy, R.A., Huang, J., Yeh, M., "Study on the use of Microsoft Kinect for robotics applications", *Position Location and Navigation Symposium (PLANS),* pp. 1280-1288, 2012.

[95] Clark, R.A., Pua, Y.H., Bryant, A.L., and Hunt, M.A., "Validity of the Microsoft Kinect for providing lateral trunk lean feedback during gait retraining", *Gait Posture*, pp. 1-3, 2013.

[96] Gabel, M., Gilad-Bachrach, R., Renshaw, E., Schuster, A., "Full body gait analysis with Kinect", *Proceedings of International Conference of the IEEE Engineering in Medicine and Biology Society IEEE Engineering in Medicine and Biology Society*, pp. 1964-1977, 2012.

[97] Stone, E.E., Skubic, M., "Passive in-home measurement of stride-to-stride gait variability comparing vision and Kinect sensing", *Proceedings of International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6491-6494, 2011.

[98] XSens, Available from: http://www.xsens.com/ (Cited: 25/09/2013)

[99] MVN, *XSens*, Available from: http://www.xsens.com/en/general/mvn/ (Cited: 25/09/2013)

[100] Mancini, M., Horak, F.B., "The relevance of clinical balance assessment tools to differentiate balance deficits", *European Journal of Physical and Rehabilitation Medicine*, vol. 46(2), pp. 239-248, 2010.

[101] Van Den Noort, J.C., Scholtes, V.A., Harlaar, J., "Evaluation of clinical spasticity assessment in cerebral palsy using inertial sensors", *Gait Posture*, vol. 30, pp. 138-143, 2009.

[102] Martínez-Ramírez, A., Lecumberri, P., Gómez, M., Rodriguez-Mañas, L., García, F. J., Izquierdo, M., "Frailty assessment based on wavelet analysis during quiet standing balance test", *Journal of Biomechanics*, vol. 44, pp. 2213-2220, 2011.

[103] Faber, G.S., Kingma, I., Bruijn, S.M., Van Dieën, J.H., “Optimal inertial sensor location for ambulatory measurement of trunk inclination”, *Journal of Biomechanics*, vol. 42(14), pp. 2406-2409, 2009.

[104] Floor-Westerdijk, M.J., Schepers, H. M., Veltink, P.H., Van Asseldonk, E. H.F., Buurke, J.H., “Use of Inertial Sensors for Ambulatory Assessment of Center of Mass Displacements During Walking”, *IEEE Transactions on Biomedical Engineering*, pp. 2080-2084, 2012.

[105] Parel, I., Cutti, A. G., Fiumana, G., Porcellini, G., Verni, G., and Accardo, A. P., “Ambulatory measurement of the scapulohumeral rhythm: intra- and inter-operator agreement of a protocol based on inertial and magnetic sensors”, *Gait Posture*, pp. 636-640, 2012.

[106] Spain, R.I., St George, R. J., Salarian, A., Mancini, M., Wagner, J.M., Horak, F.B., Bourdette, D., “Body-worn motion sensors detect balance and gait deficits in people with multiple sclerosis who have normal walking speed” *Gait Posture*, vol. 35, pp. 573-578, 2012.

[107] IGS 180, *Animazoo*, Available from: http://www.animazoo.com/products/igs-180-range (Cited: 25/09/2013)

[108] Gypsy 7 and eMove, picture courtesy of eMove project and Animazoo.

[109] MPU9150 Datasheet, *InvenSense*, Available from: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/IMU/PS-MPU-9150A.pdf (Cited: 25/09/2013)

[110] Aar, C., Shkel, A., “MEMS Vibratory Gyroscopes: Structural Approaches to Improve Robustness”, October 2010, ISBN: 978-1441934895.

[111] Kaajakari, V., “Practical MEMS: Analysis and Design of Microsystems”, October 2010, ISBN: 978-0982299104.

[112] Qin, F., Xu, J., Jiang, S., “A New Scheme of Gyroscope Free Inertial navigation System Using 9 Accelerometers”, *International Workshop on Intelligent Systems and Applications (ISA)*, pp. 1-4, 2009.

[113] Tsai, Y.L., Tu, T.T., Chou, P.H., “EcoIMU: A compact, wireless, gyro-free inertial measurement unit based on two triaxial accelerometers”, *Proceedings of the 10th ACMIEEE International Conference on Information Processing in Sensor Networks*, pp. 133-134, 2011.

[114] Brown, R.G., Hwang P.Y.C., “Introduction to Random Signals and Applied Kalman Filtering”, ISBN: 978-0471128397.

[115] Yun, X.Y.X., Aparicio, C., Bachmann, E.R., McGhee, R.B., “Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking”, *IEEE Transactions on Robotics*, pp. 1216-1227, 2006.

[116] Nguyen, K.D., Cheng, I.M., Luo, Z., Yeo, S.H., Duh, H.B., “A body sensor network for tracking and monitoring of functional arm motion”, *Proceedings*

of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3862-3867, 2009.

[117] Yang, G.Z., Yacoub, M., “Body Sensor Networks”, ISBN: 978-1846282720.

[118] Van Laerhoven, K., Kern, N., Gellersen, H.W., Schiele, B., “Towards a wearable inertial sensor network”, *IEEE Eurowearable*, pp. 125-130, 2003.

[119] Lorincz, K., Chen, B., Challen, G.W., Chowdhury, A.R., Patel, S., Bonato, P., Welsh, M., “Mercury: a wearable sensor network platform for high-fidelity motion analysis”, *Energy*, pp. 183-196, 2009.

[120] Otto, C., Milenković, A., Sanders, C., Jovanov, E., “System architecture of a wireless body area sensor network for ubiquitous health monitoring”, *Journal of Mobile Multimedia*, vol. 1(4), pp. 307-326, 2006.

[121] Smart-its, Available from: http://eis.comp.lancs.ac.uk/DiySmartits/ (Cited: 08/02/2014)

[122] Van Laerhoben, K., Schmidt, A., Gellersen, H., “Multi-Sensor Context-Aware Clothing”, *Proceedings of the 6th International Symposium on Wearable Computers (ISWC)*, pp. 49-57, 2002.

[123] Maths Rotations, *Euclidean Space*, Available from: http://www.euclideanspace.com/maths/geometry/rotations/ (Cited: 25/09/2013)

[124] Kovar, L., Schreiner, J., Gleicher, M., “Footskate cleanup for motion capture editing”, *Proceedings of the ACM SIGGRAPH Eurographics Symposium on Computer Animation (SCA)*, 2002.

[125] Gilles, B., Moccozet, L., Magnenat-Thalmann, N., “Anatomical modelling of the musculoskeletal system from MRI”, *Medical Image Computing and Computer-Assisted Intervention*, pp. 289-296, 2006.

[126] Chao, E.Y., Armiger, R.S., Yoshida, H., Lim, J., Haraguchi, N., “Virtual interactive musculoskeletal system (VIMS) in orthopaedic research, education and clinical patient care”, *Journal of Orthopaedic Surgery and Research,* 2007.

[127] Young, A.D., “From posture to motion: the challenge for real time wireless inertial motion capture”, *Proceedings of 5th International Conference on Body Area Networks*, 2010.

[128] Lander, J., “Working With Motion Capture File Formats”, *Game Developer*, Available online: http://www.darwin3d.com/gamedev/articles/col0198.pdf (Cited: 25/09/2013)

[129] Meredith, M., Maddock, S., “Motion Capture File Formats Explained”, Available online: http://www.dcs.shef.ac.uk/intranet/research/public/resmes/CS0111.pdf (Cited: 25/09/2013)

[130] Bodenheimer, B., Rose, C., Rosenthal, S., Pella, J., "The Process of Motion Capture: Dealing with the Data", *Computer Animation and Simulation*, Eurographics, pp. 3-18, 1997.

[131] Chung, H.S., Lee, Y., "MCML: motion capture markup language for integration of heterogeneous motion capture data", *Computer Standards & Interfaces*, vol. 26(2), pp. 113-130, 2004.

[132] 3ds Max, *Autodesk*, Available from: http://www.autodesk.com/products/autodesk-3ds-max/overview (Cited: 25/09/2013)

[133] MotionBuilder, *Autodesk*, Available from: http://www.autodesk.com/products/motionbuilder/overview (Cited: 25/09/2013)

[134] Maya, *Autodesk*, Available from: http://www.autodesk.com/products/autodesk-maya/overview (Cited: 25/09/2013)

[135] BVHImportExport, *Creative Crash*, Available from: http://www.creativecrash.com/maya/ plugin/bvh-file-import-export-for-maya (Cited: 25/09/2013)

[136] AMC2MOV and ASF2MEL converters, *Creative Crash*, Available from: http://www.creativecrash.com/maya/downloads/applications/3d-converters/c/asf2mel-and-amc2mov-sgi (Cited: 25/09/2013)

[137] Blender, Available from: http://www.blender.org/ (Cited: 25/09/2013)

[138] Life Forms Studio, *Credo Interactive*, Available from: http://www.credo-interactive.com/ products/lifeforms/ (Cited: 25/09/2013)

[139] Poser, *Smith Micro Software*, Available from: http://poser.smithmicro.com/ (Cited: 25/09/2013)

[140] Carrara, *Daz3D*, Available from: http://www.daz3d.com/products/carrara/carrara-what-is-carrara/ (Cited: 25/09/2013)

[141] QT, *QT Project*, Available from: http://qt-project.org (Cited: 25/09/2013)

[142] Signals and Slots, *QT Project*, Available from: http://qt-project.org/doc/qt-4.8/signalsandslots.html (Cited: 25/09/2013)

[143] Challis, J.H., "Precision of the estimation of human limb inertial parameters", *Journal of Applied Biomechanics*, vol. 15(4), 1999, 418-428.

[144] Shimmer, Available from: http://www.shimmersensing.com (Cited: 20/01/2014)

[145] MSP430 Microcontroller Documentation, *Texas Instruments*, Available from: http://www.ti.com/lsds/ti/microcontroller/16-

bit_msp430/overview.page?DCMP=MCU_other&HQS=msp430 (Cited: 25/09/2013)

[146] IDG 500 Gyroscope Documentation, *InvenSense*, Available from: http://www.invensense.com/mems/gyro/documents/PS-IDG-0500B-00-08.pdf (Cited: 25/09/2013)

[147] ADXL345 Accelerometer Documentation, *Analog Devices*, Available from: http://www.analog.com/en/mems-sensors/mems-inertial-sensors/adxl345/products/product.html (Cited: 25/09/2013)

[148] HMC5843 Magnetometer Documentation, *Honeywell*, Available from: http://www.honeywell.com/sites/servlet/com.merx.npoint.servlets.DocumentServlet?docid=DA9ACFE3C-F7C0-9998-6085-D9D84941499D (Cited: 25/09/2013)

[149] Wii Remote Plus, *Nintendo*, Available from: http://www.nintendo.co.uk/Wii/Accessories/Accessories-Wii-Nintendo-UK-626430.html (Cited: 20/01/2014)

[150] Shimmer Motion Development Kit, *Shimmer*, Available from: http://www.shimmersensing.com/shop/motion-development-kit (Cited: 20/01/2014)

[151] Razor AHRS IMU Documentation, *SparkFun*, Available from: https://www.sparkfun.com/products/10736 (Cited: 25/09/2013)

[152] FTDI Basic Breakout Board Documentation, *SparkFun*, Available from: https://www.sparkfun.com/products/9716 (Cited: 25/09/2013)

[153] ITG3200 Gyroscope Documentation, InvenSense, Available from: http://invensense.com/mems/gyro/itg3200.html (Cited: 25/09/2013)

[154] HMC5883L Magnetometer Documentation, *Honeywell*, Available from: http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf (Cited: 25/09/2013)

[155] IMU3000 Gyroscope Documentation, *InvenSense*, Available from: http://www.invensense.com/mems/gyro/imu3000.html (Cited: 25/09/2013)

[156] MMA8452Q Accelerometer Documentation, *Freescale*, Available from: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MMA8452Q (Cited: 25/09/2013)

[157] 8-bit AVR, *Atmel*, http://www.atmel.com/products/microcontrollers/avr/default.aspx (Cited: 25/09/2013)

[158] MPU9150, *InvenSense*, http://www.invensense.com/mems/gyro/mpu9150.html (Cited: 25/09/2013)

[159] Perani ESD100, *Sena*, http://www.sena.com/download/manual/manual_parani_esd-v1.1.4.pdf (Cited: 22/01/2014)

[160] Shimmer 3, *Shimmer*, http://www.shimmersensing.com/shop/shimmer3-development-kit (Cited: 20/01/2014)

[161] Ballagas, R., Rohs, M., Sheridan, J., Borchers, J., "The Smart Phone: A Ubiquitous Input Device", *IEEE Pervasive Computing*, vol. 5(1), 2006.

[162] Ganti, R.K., Srinivasan, S., Gacic, A., "Multisensor Fusion in Smart- phones for Lifestyle Monitoring", *Proceedings of International Conference on Body Sensor Networks*, pp. 36-43, 2010.

[163] Keally, M., Zhou, G., Xing, G., Wu, J., Pyles, A., "PBN: Towards Practical Activity Recognition Using Smartphone-Based Body Sensor Networks", *Computer Engineering*, ACM, pp. 246-259, 2011.

[164] Turner, H., White, J., "Verification and Validation of Smartphone Sensor Networks", *MobilWare*, 2011.

[165] Yamada, M., Aoyama, T., Mori, S., Nishiguchi, S., Okamoto, K., Ito, T., Muto, S., Ishihara, T., Yoshitomi, H., Ito, H., "Objective Assessment of Abnormal Gait in Patients With Rheumatoid Arthritis Using a Smartphones", *Rheumatology International*, pp. 1-6, 2011.

[166] Hebden, L., "Development of Smartphone Applications for Nutrition and Physical Activity Behavior Change", *JMIR Research Protocols*, vol. 1(2), 2012.

[167] Colwell, B., "Engineers, Programmers, and Black Boxes", *Computer*, vol. 38(3), pp. 8-11, 2005

[168] Kotonya, G., "An Architecture-Centric Development Environment for Black-Box Component-Based Systems", *Software Architecture (Springer Berlin Heidelberg)*, pp. 98-113, 2008.

[169] Lozano, C., Jindrich, D., Kahol, K., Mall, E.T., "The Impact on Musculoskeletal System during Multitouch Tablet Interactions". *Interfaces*, pp. 825-828, 2011.

[170] OpenGL ES, *Khronos*, Available from: http://www.khronos.org/opengles/ (Cited: 01/10/2013)

[171] Olfati-Saber, R., "Distributed Kalman Filtering and Sensor Fusion in Sensor Networks", *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[172] Ayub, S., Bahraminasab, A. Honary, B., "A Sensor Fusion Method for Smart phone Orientation Estimation", *Proceedings of 13th Annual Post Graduate Symposium on the Convergence of Telecommunications*, Networking and Liverpool, United Kingdom, 2012.

[173] FitBit, Available from: http://www.fitbit.com (Cited: 01/10/2013)

[174] Up Wristband, *Jawbone*, Available from: https://jawbone.com/up (Cited: 01/10/2013)

[175] S-Health, *Samsung*, Available from: http://www.samsung.com/uk/news/localnews/2012/samsung-introduces-s-health-application-for-galaxy-s-iii (Cited: 01/10/2013)

[176] Tomlein, M., Bielik, P., Kratky, P., Mitrk, S., Barla, M., Bielikova, M., "Advanced Pedometer for Smartphone-Based Activity Tracking", *Proceeding of the International Conference on Health Informatics*, SciTe Press, pp. 401-404, 2012.

[177] Tacconi, C., Mellone, S., Chiari, L., "Smartphone-based applications for investigating falls and mobility", *Proceedings of 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pp. 258-261, 2011.

[178] Ghose, A., "Road Condition Monitoring and Alert Application: Using In-Vehicle Smartphone as Internet-Connected Sensor", *Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 489-491, 2012.

[179] Google Charts, *Google*, Available from: https://developers.google.com/chart/ (Cited: 27/01/2014)

[180] WebGL, *Khronos*, Available from: http://www.khronos.org/webgl/ (Cited: 01/10/2013)

[181] Cortex A50 Series, *ARM*, Available from: http://www.arm.com/products/processors/cortex-a50/index.php (Cited: 01/10/2013)

Appendix A – Computing Quaternion Algebra

At the software environment level, inertial motion capture relies on geometric calculations comprising of both rotational and positional data structures. This section introduces the fundamental geometry driving the software environment and the mobile application's back-end. The software environment features two bespoke libraries, entitled Vector and Quaternion, containing all relevant mathematical functionality. The Vector library delineates the software environment's most basic data structure: a linear three-valued matrix accompanied by a specific set of mathematical operations. A vector object may define a viewing direction, a translation transformation, a point in space or an Euler rotation. The Quaternion library is used to represent rotation and compute rotational operations. A quaternion is a four-dimensional (4D) space vector defined by a three-dimensional axis of rotation and a scalar transformation along that axis. The quaternion rotational model is deployed, instead of the Euler equivalent, to avoid the gimbal lock problem. Because trigonometry is computationally expensive, most software environments adopt this approach for performance reasons. The following formula illustrates these two elementary data structures where v is a vector, q is a quaternion, $v' \in q$ and s is the scalar value.

$$v = [\,x \quad y \quad z\,] \qquad\qquad q = [\,s \quad v'\,]$$

Motion data is stored as Euler rotations. An initial conversion is required for Skeletrix to decipher and store rotational transformations. The following formula utilizes many trigonometric operations that compromise performance. In the following example, vector $[\,x \quad y \quad z\,]$ is converted into quaternion $[\,s \quad v_x \quad v_y \quad v_x\,]$.

$$s = \left(\cos\frac{x}{2} * \cos\frac{y}{2} * \cos\frac{z}{2}\right) - \left(\sin\frac{x}{2} * \sin\frac{y}{2} * \sin\frac{z}{2}\right)$$

$$v_x = \left(\cos\frac{x}{2} * \cos\frac{y}{2} * \sin\frac{z}{2}\right) + \left(\sin\frac{x}{2} * \sin\frac{y}{2} * \cos\frac{z}{2}\right)$$

$$v_y = \left(\sin\frac{x}{2} * \cos\frac{y}{2} * \sin\frac{z}{2}\right) + \left(\cos\frac{x}{2} * \sin\frac{y}{2} * \cos\frac{z}{2}\right)$$

$$v_x = \left(\cos\frac{x}{2} * \sin\frac{y}{2} * \cos\frac{z}{2}\right) - \left(\sin\frac{x}{2} * \cos\frac{y}{2} * \sin\frac{z}{2}\right)$$

The opposite conversion, quaternion to Euler, has two purposes. First, Euler results are required when outputting valid motion capture files (e.g. BVH or BVHE). Second, all rotations must displayed comprehensibly throughout the interface layer (in accordance to the principle of transparency introduced in the requirements specification). The following formula, implemented in the Quaternion library, transforms quaternion $[\,s \quad v_x \quad v_y \quad v_x\,]$ into resulting vector r. The result is in radians.

$$r_x \;=\; \tan^{-1}\frac{(2 * v_y * s) - (2 * v_x * v_z)}{1 - 2 * {v_y}^2 - 2 * {v_z}^2}$$

$$r_y \;=\; \sin^{-1}(2 * v_x * v_y) + (2 * v_z * s)$$

$$r_x \;=\; \tan^{-1}\frac{(2 * v_x * s) - (2 * v_y * v_z)}{1 - 2 * {v_x}^2 - 2 * {v_z}^2}$$

When applying motion data to its skeleton, the kinematic model computes numerous sequences of rotations. Those sequences employ quaternion multiplication, an associative yet non-commutative operation. The quaternion product is used to compute the blended result of two rotations. It uses computationally inexpensive mathematical operations such as multiplication, addition and subtraction. Given that Skeletrix performs thousands of multiplications every second, each computation must be conducted efficiently. Division, as needed during calibration when estimating compensational differences, is calculated by inverting one of the quaternions before applying the product formula. In the following example, r stores the blended result of q and q'. Once completed, all quaternion products must be normalized.

$$r_s \;= q_s * {q_s}' - \; q_{v_x} * {q_{v_x}}' - \; q_{v_y} * {q_{v_y}}' - \; q_{v_z} * {q_{v_z}}'$$

$$r_{v_x} \;=\; q_s * {q_{v_x}}' + q_{v_x} * {q_s}' + q_{v_y} * {q_{v_z}}' - \; q_{v_z} * {q_{v_y}}'$$

$$r_{v_y} = q_s * {q_{v_y}}' - q_{v_x} * {q_{v_z}}' + \; q_{v_y} * {q_s}' + \; q_{v_z} * {q_{v_x}}'$$

$$r_{v_z} = \; q_s * {q_{v_z}}' + q_{v_x} * {q_{v_y}}' - q_{v_y} * {q_{v_x}}' + \; q_{v_z} * {q_s}'$$

Aside from rotation sequences, there are many situations that require a joint effort between the Vector and Quaternion libraries. As previously mentioned, the rendering engine calculates all topology at the software environment level and not through the OpenGL pipeline. Each bone's geometry is instantiated in its default form before becoming subject to a series of transformations. Those transformations can be rotations, size adjustments or translations. When the skeleton moves, its constituent topologies rotate in accordance to a set of kinematic constraints. The following equation is relevant in that scenario. It rotates point p, in local space, by quaternion q to generate the resulting vector r.

$$\begin{aligned} r_x = {} & (2 * q_{v_x} * q_{v_y} * p_x) + ({q_{v_y}}^2 * p_z) + (2 * q_{v_z} * \ v_y * p_y) + (2 * q_s * q_{v_z} * p_x) \\ & - ({q_{v_z}}^2 * p_z) + \ (s{q_s}^2 * p_z) \ - (2 * q_{v_x} * q_s * p_y) \ - ({q_{v_x}}^2 * p_z) \end{aligned}$$

$$\begin{aligned} r_y = {} & (2 * q_{v_x} * q_{v_z} * p_x) + (2 * q_{v_y} * q_{v_z} * p_z) + ({q_{v_z}}^2 * p_y) - (2 * q_s * q_{v_y} * p_x) \\ & - ({q_{v_y}}^2 * p_y) + (2 * q_s * q_{v_x} * p_z) - ({q_{v_x}}^2 * p_y) + ({q_s}^2 * p_y) \end{aligned}$$

$$\begin{aligned} r_z = {} & ({q_s}^2 * p_x\ ) + (2 * \ q_{v_y} * q_s * p_y) - (2 * \ q_{v_z} * q_s * p_z) + ({q_{v_x}}^2 * p_x) + (2 \\ & * q_{v_y} * q_{v_x} * p_z) + (2 * q_{v_z} * q_{v_x} * p_y) - ({q_{v_z}}^2 * p_x) - ({q_{v_y}}^2 * p_x) \end{aligned}$$

There are several circumstances where vectors can represent gyrations. For example, a vector can be represented as a line segment starting at the origin and ending at a specific point in space. A second line segment v', of length $|\ v\ |$, is drawn in the default position: also starting at the origin but pointing upwards. In this circumstance, a rotation can be used to describe the transformation required for line segment v' to assume the orientation of v. This equation shows how to convert vector positions into rotations. Vector v' is defined as follows.

$$v' = [0 \quad |v| \quad 0]$$

The subsequent equation is applied to conclude the desired rotation:

$$r = \frac{v' \times v * \sqrt{\frac{v' \times v}{2 * |v'| * |v|}}}{|v' \times v|}$$

Appendix B – DDK Driver for Razor AHRS

B.1 Razor Global Header

The *Razor_global* class Defines the header of the DLL.

Razor_global.h

```
/*************************************************************************/
/*                           Razor_global Header                         */
/*************************************************************************/

#ifndef RAZOR_GLOBAL_H
#define RAZOR_GLOBAL_H

#include <QtCore>

#if defined(RAZOR_LIBRARY)
#  define RAZORSHARED_EXPORT Q_DECL_EXPORT
#else
#  define RAZORSHARED_EXPORT Q_DECL_IMPORT
#endif

#endif
```

B.2 Razor Class

The *Razor* class contains all the code required to package data from the BSN.

Razor.h

```
/*************************************************************************/
/*                           Razor Class Header                          */
/*************************************************************************/

#ifndef RAZOR_H
#define RAZOR_H
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <cstdio>
#include <vector>
#include <windows.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <iomanip>
#include <sstream>
#include <string>
#include <QThread>
#include "Razor_global.h"
#include "Sensor.h"

using namespace std;

class RAZORSHARED_EXPORT Razor {
public:
    void dllAction1();
    void dllAction2();
    void dllAction3();
    void dllAction4();
    void setPort(vector<int>);
    vector<float> getStream();
private:

};

#endif /* RAZOR_H */
```

Razor.cpp

```
/****************************************************************************/
/*                                 Driver Class                             */
/****************************************************************************/

#include "Razor.h"
#include <QThread>
#include "Razor_global.h"
#include "Sensor.h"
Sensor* sensor = new Sensor();

extern "C" __declspec(dllexport) void dllAction1()
{
    sensor->action1();
}

extern "C" __declspec(dllexport) void dllAction2()
{
    sensor->action2();
}

extern "C" __declspec(dllexport) void dllAction3()
{
    sensor->action3();
}

extern "C" __declspec(dllexport) void dllAction4()
{
   QThread* thread = new QThread;
   sensor->moveToThread(thread);
   thread->connect(thread, SIGNAL(started()), sensor, SLOT(update()));
   thread->start();
}

extern "C" __declspec(dllexport) void setPort(int p)
{
    sensor->setPort(p);
}

extern "C" __declspec(dllexport) vector<float> getStream()
{
    sensor->update();
    return sensor->stream;
}
```

B.3 Sensor Class

The *Sensor* class contains all the code required to package data from the BSN.

Sensor.h

```
/*************************************************************************/
/*                          Sensor Class Header                          */
/*************************************************************************/

#ifndef SENSOR_H
#define SENSOR_H

#include <QObject>
#include <vector>
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <cstdio>
#include <vector>
#include <windows.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <iomanip>
#include <sstream>
#include <string>
#include <QThread>

using namespace std;

class Sensor : public QObject
{
   Q_OBJECT

public:
    int port, colon1, colon2, end;
    char r[97];
    vector<float> stream;
    HANDLE h;
    DWORD writemsg;
    DWORD readmsg;
    string message;
    Sensor(QObject* parent = 0);
    void action1();
    void action2();
    void action3();
    void action4();
    void setPort(int);
    std::vector<float> getStream();
    void unpack();
public Q_SLOTS:
   void update();
};

#endif /* SENSOR_H */
```

Sensor.cpp

```
/**************************************************************************/
/*                              Sensor Class                              */
/**************************************************************************/

#include "Sensor.h"

/* Constructor. */
Sensor::Sensor(QObject* parent) : QObject(parent) {
    stream.push_back(0);
    stream.push_back(0);
    stream.push_back(0);
    port = 0;
}

/* Driver action 1 that connects to the port. */
void Sensor::action1() {
    writemsg = 1;
    readmsg = 0;
    message = "";
    std::wstring port_prefix = L"\\\\.\\COM";
    std::wostringstream int_to_wstring;
    int_to_wstring << port;
    std::wstring port_value = port_prefix + int_to_wstring.str();
    h =
CreateFile(port_value.c_str(),GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXISTIN
G,0,NULL);
    if(h != INVALID_HANDLE_VALUE) {
        DCB dcb;
        memset(&dcb,0,sizeof(dcb));
        dcb.DCBlength = sizeof(dcb);
        dcb.BaudRate = 57600;
        dcb.fBinary = 1;
        dcb.fDtrControl = DTR_CONTROL_DISABLE;
        dcb.fRtsControl = RTS_CONTROL_DISABLE;
        dcb.Parity = NOPARITY;
        dcb.StopBits = ONESTOPBIT;
        dcb.ByteSize = 8;
    }
}

/* Driver action 2. */
void Sensor::action2() {
}

/* Driver action 3. */
void Sensor::action3() {
}

/* Driver action 4. */
void Sensor::action4() {
}

/* Set communication port. */
void Sensor::setPort(int p) {
    port = p;
}

/* Returns stream of data. */
vector<float> Sensor::getStream() {
    return stream;
}

/* Parses the incoming data from the sensor. */
void Sensor::update() {
    if(ReadFile(h,r,sizeof(r),&readmsg,NULL)){
         colon1 = 0;
         colon2 = 0;
         end = 0;
         string received(r);
         if(received.substr(0,5) == "#YPR="){
             for(unsigned int i = 5; i < received.length(); i++){
                 if(received.at(i) == '\n') {
                     end = i;
                     break;
                 }
             }
             message = received.substr(5,end-5);
             for(unsigned int j = 0; j < message.length(); j++){
                 if(message.at(j) == ',') {
                     if(colon1 == 0) {
                         colon1 = j;
                     }else{
                         colon2 = j;
                         break;
                     }
                 }
             }
             if(colon1 != 0 && colon2 != 0) {
                  stream[0] =
::atof(message.substr(0,colon1).c_str())/57.2957795f;
                  stream[1] = ::atof(message.substr(colon1+1,colon2-1 -
colon1+1).c_str())/57.2957795f;
                  stream[2] =
::atof(message.substr(colon2+1,message.length() -
colon2+1).c_str())/57.2957795f;
             }
         }
     }
}
```

Appendix C – MotionCloud API JSON Packets

C.1 Profile

This is an example JSON packet of what *profile_get* API call returns.

Example Profile JSON Packet

```
{
        "username" : "tpascu1",
        "password" : "password1",
        "password_confirm" : "password1",
        "firstname" : "Tudor",
        "lastname" : "Pascu",
        "email" : "t.pascu@sussex.ac.uk"
}
```

C.2 Group

This is an example JSON packet of what *group_get* API call returns.

Example Group JSON Packet

```
{
        "group_id": "20",
        "title": "Smartphone Group 1",
        "description": "7 smartphone motion recording.",
        "recordings": [
                {
                        "recording_id": "50",
                        "name": "Smartphone Recording 1",
                        "channels": [ // channel data omitted ]
                },
                {
                        "recording_id": "51",
                        "name": "Smartphone Recording 2",
                        "channels": [ // channel data omitted ]
                },
                {
                        "recording_id": "52",
                        "name": "Smartphone Recording 3",
                        "channels": [ // channel data omitted ]
                },
                {
                        "recording_id": "53",
                        "name": "Smartphone Recording 4",
                        "channels": [ // channel data omitted ]
                },
                {
                        "recording_id": "54",
                        "name": "Smartphone Recording 5",
                        "channels": [ // channel data omitted ]
                },
                {
                        "recording_id": "55",
                        "name": "Smartphone Recording 6",
                        "channels": [ // channel data omitted ]
                },
                {
                        "recording_id": "56",
                    "name": "Smartphone Recording 7",
                    "channels": [ // channel data omitted ]
              },

       ]
}
```

This is an example JSON packet of what *group_getlist* API call returns.

Example Recording JSON Packet

```
{
       "ids": [
                    "20",
                    "21",
                    "22",
                    "23",
                    "24",
                    "25",
                    "26"
             ],
       "titles": [
                    "Smartphone Group 1",
                    "Smartphone Group 2",
                    "Smartphone Group 3",
                    "Smartphone Group 4",
                    "Software Environment Recordings 1",
                    "Software Environment Recordings 2",
                    "Software Environment Recordings 3"

             ],
       "descriptions": [
                    "7 smartphone motion recording.",
                    "5 smartphone motion recording.",
                    "5 smartphone motion recording.",
                    "Activity data uploaded from smartphone.",
                    "Recording Uploaded From Software Environment",
                    "Test Description of Smartphone Recording 2.",
                    "Test Description of Smartphone Recording 3."
             ]
}
```

C.3 Recording

This is an example JSON packet of what *recording_get* API call returns.

Example Recording JSON Packet

```
{
       "title": "Suit Recording",
       "description": "7 channels of data",
       "channels": [
             {
                    "channel_id": "1820",
                    "name": "Spine",
                    "vectors": [ // channel data omitted ]
             },
             {
                    "channel_id": "1821",
                    "name": "Left_Arm",
                    "vectors": [ // channel data omitted ]
```

```
            },
            {
                  "channel_id": "1822",
                  "name": "Right_Arm",
                  "vectors": [ // channel data omitted ]
            },
            {
                  "channel_id": "1823",
                  "name": "Left_Forearm",
                  "vectors": [ // channel data omitted ]
            },
            {
                  "channel_id": "1824",
                  "name": "Right_Forearm",
                  "vectors": [ // channel data omitted ]
            },
            {
                  "channel_id": "1825",
                  "name": "Left_Hand",
                  "vectors": [ // channel data omitted ]
            },
            {
                  "channel_id": "1826",
                  "name": "Right_Hand",
                  "vectors": [ // channel data omitted ]
            }
      ]
}
```

This is an example JSON packet of what *recording_getlist_uid* API call returns.

Example Recording JSON Packet

```
{
      "ids": [
                  "24",
                  "25",
                  "64",
                  "66",
                  "67"
            ],
      "titles": [
                  "Suit Test Recording 1",
                  "Suit Test Recording 1",
                  "Android Recording 1",
                  "Android Recording 2",
                  "Android Recording 3"
            ],
      "descriptions": [
                  "Test Description 1.",
                  "Test Description 2.",
                  "Test Description of Smartphone Recording 1.",
                  "Test Description of Smartphone Recording 2.",
                  "Test Description of Smartphone Recording 3."
            ]
}
```

C.4 Channel

This is an example JSON packet of what *channel_get* API call returns.

Example Channel JSON Packet

```
{
	"channel_id": "1820",
	"name": "Spine",
	"vectors": [
		{
			"vector_id": "565019",
			"channel_id": "1820",
			"x": "128.388",
			"y": "-10.456,
			"z": "8.844"
		},
		{
			"vector_id": "565020",
			"channel_id": "1820",
			"x": "129.618",
			"y": "-10.691",
			"z": "8.422"
		},
		{
			"vector_id": "565021",
			"channel_id": "1820",
			"x": "130.745",
			"y": "-10.569",
			"z": "8.474"
		},
		{
			"vector_id": "565022",
			"channel_id": "1820",
			"x": "132.898",
			"y": "-10.850",
			"z": "8.859"
		},
		// values omitted
	]
}
```

This is an example JSON packet of what *channel_getlist* API call returns.

Example Recording JSON Packet

```
{
	"ids": [
			"1820",
			"1821",
			"1822",
			"1823",
			"1824",
			"1825",
			"1826"
		],
	"names": [
			"Spine",
			"Left_Arm",
			"Right_Arm",
			"Left_Forearm",
			"Right_Forearm",
			"Left_Hand",
                    "Right_Hand"
             ]
}
```

C.5 Vector

This is an example JSON packet of what *vector_get* API call returns.

Example Vector JSON Packet

```
{
      "vector_id": "565019",
      "channel_id": "1729",
      "x": "128.388",
      "y": "-10.456,
      "z": "8.844"
}
```